SDN MAGAZINE

Nummer
144
Okt 2022

SDN Magazine is
een uitgave van:
SDN

Van en voor Microsoft & .NET professionals



> Teams apps ontwikkelen met Dataverse integratie

> CoderDojo en Astro PI

GEORGIA CODES TO HELP STUDENTS INTO THEIR DREAM EDUCATION

What do you want to code for?



Georgia helped to develop a one-stop communication platform for students to store important, personal information. Ultimately, the improved system grants students easier access to enrol or re-enrol at a college or university, making it easier to pursue their dream careers!

How do you want to make a difference?
Have a look at our vacancies:

netcompany

Colofon

Uitgave
Software Development Network
26ste jaargang
Nr. 144 • oktober 2022

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

Auteurs:
Dibran Mulder, Jan de Vries, Kornelis Hietkamp, Jessy The, Jelle de Graaf, Koen Aerts, Sanneke van der Meer, Menno Jongerius, Vincent Hendriks en Jasper Sprengers.

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

©2022 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-ers,

Het lijkt alweer een hele tijd geleden, maar wat hebben we een mooie Future Tech conferentie gehad gevolgd door een prachtige zomer.

Als partner van ook andere events/conferenties zoals TEQnation en natuurlijk onze eigen Future Tech, al 2 magazine edities en een hoop leuke sprekers die op bezoek kwamen bij de SDN-cast heeft SDN een mooi voorseizoen achter de rug. Maar het jaar is nog niet om! En er zal dus nog veel moois aankomen.

De zomer zit erop, de blaadjes vallen van de bomen, het regent bijna non-stop. Dus dat betekend dat het cozy seizoen weer voor de deur staat! En hoe kan deze periode beter begonnen worden dan met een nieuwe editie van het SDN magazine?

In deze editie zullen wij terugblikken op een mooi en succesvol Future Tech, waarvan wij hopen dat jullie hier net zo van genoten hebben als wij. Maar we hebben ook weer super leuke en interessante artikelen van onze auteurs. Al met al weer een mooie nieuwe editie naar mijn mening.

Bij deze wil ik dan ook graag de auteurs bedanken die deze editie mede-mogelijk gemaakt hebben door hun bijdrage: Dibran Mulder, Jan de Vries, Kornelis Hietkamp, Jessy The, Jelle de Graaf, Koen Aerts, Sanneke van der Meer, Menno Jongerius, Vincent Hendriks en Jasper Sprengers.

Wil jij ook jouw kennis delen met de SDN community? Dan ben ook jij natuurlijk van harte welkom om jouw artikel in te sturen! Stuur deze dan naar redactie@sdn.nl en wie weet sta jij in de volgende editie van het magazine!

Voor nu, veel leesplezier!

Kelly Verschoor
kelly.verschoor@sdn.nl



DEVITJOBS.NL

Senior Software Developer

Salarisindicatie:
45.000 - 65.000 EUR

Locatie:
Delft

Technologieën:
C#, Java, PHP, Azure, Kubernetes, Docker, JavaScript, Angular, NodeJS, React, .NET

netcompany

Senior Software Developer

Salarisindicatie:
45.000 - 75.000 EUR

Locatie:
Delft

Technologieën:
Java, C++, C#, Cloud, DevOps, Docker, Vue, React, Angular, JavaScript

TOPdesk
Your guides to service excellence

Azure DevOps Engineer

Salarisindicatie:
60.000 - 85.000 EUR

Locatie:
Nieuwegein

Technologieën:
Azure, Power BI, PowerShell, C#, DevOps, CI/CD, .NET

ORDINA
Software Development

Medior .NET developer

Salarisindicatie::
45.000 - 60.000 EUR

Locatie:
Amstelveen

Technologieën:
C#, MVC, .NET Core, Angular, JavaScript, HTML5, CSS, CI/CD, Git, Jenkins, Azure, JIRA, Confluence, Kubernetes, Docker

Atos

Find all these vacancies and more

at Devitjobs.nl

In dit nummer

06 Zero-trust Cloud Architectuur
10 Teams apps ontwikkelen met Dataverse integratie
16 “Zorg-ICT is de ideale sector, omdat ik graag mensen help”
18 5 reasons waarom je in blockchain moet verdiepen als frontend developer
24 Future Tech terugblik
26 Slim automatiseren vanuit één centrale plek in de organisatie
28 Why we suck at estimation
30 LITTL - Deel jouw IT kennis met kinderen!
32 CoderDojo en Astro PI
34 Modulaire opzet in Blazor Webassembly
38 Een Mobile App zonder UI (logica)
42 Agile 2: meer dan een upgrade



10 Veel lezers zullen wel bekend zijn met een van de populaire front-end frameworks/libraries, zoals Angular, React, Vue.js, Blazor of een van de vele andere. De manier om hier snel professionele oplossingen in te ontwikkelen is ook flink verbeterd in de afgelopen jaren.

18 Wist je dat je met frontend skills kan bijdragen aan Web3 en de Metaverse? Misschien denk je: “maar dat heeft toch met blockchain te maken”? Zeker! Blockchain is een van dé bouwstenen van Web3 en de Metaverse. Web3 is een idee van de (opkomende!) derde versie van het internet waarbij blockchain wordt gebruikt voor de data van users. De Metaverse is een alternatieve virtuele wereld waarbij in combinatie met Web3 mensen veilig kunnen connecten, spelen en werken in deze virtuele werelden door middel van blockchain.

34 Nu Blazor steeds prominenter in het Microsoft landschap terechtkomt, zie ik de eerste teams productie applicaties opzetten in Blazor. Eén van de uitdagingen die je daarbij hebt, is hoe je de nieuwe applicatie structureert, zodat hij kan groeien zonder dat het een chaotische brij wordt. In de backend systemen hebben we tal van mogelijkheden om applicaties op te knippen. In de front-end zijn die opties lastiger. Ik hoop in dit artikel een paar handvatten te geven om je Blazor client modulair op te zetten.

Zero-trust Cloud Architectuur

Auteur: Dibran Mulder

Met de komst van de public cloud is het IT-landschap van veel organisaties sterk veranderd. Tegenwoordig hebben organisaties vaak meerdere landing zones in bijvoorbeeld de public cloud aangevuld met on-premises landing zones of infrastructuur die op de edge draait. Deze diversificatie van het IT-landschap die nog eens versterkt werd door het aantal agents die deelnemen aan het netwerk van een organisatie, denk daarbij aan mobiele telefoons, IoT devices en dergelijke, heeft ervoor gezorgd dat organisaties anders naar het security vraagstuk van hun IT-landschap zijn gaan kijken.
Zero trust wordt vaak ook wel perimeterless security genoemd, wat eigenlijk de lading best goed dekt.
In het pre cloud tijdperk werd er veel in lagen van security gedacht, eigenlijk net zoals vroeger bij een kasteel. De slotgracht was de eerste perimeter (laag) van beveiliging, de kasteelmuur was de volgende perimeter en tot slot was er de burcht van het kasteel waar de meest waardevolle spullen werden opgeborgen. Dit werkte eigenlijk best goed totdat de diversificatie optrad. Het IT-landschap werd eigenlijk meer en meer een mesh van landing zones, applicaties, devices en users die in verschillende configuraties met elkaar moeten communiceren. En daarmee voldoet het traditionele lagen model niet meer aan de behoefte van deze tijd en moeten we andere concepten introduceren om het IT Landschap veilig te houden.

Zero trust architectuur

Er valt veel te zeggen over zero trust architectuur, het domein is vrij complex en daarom is het voor developers soms moeilijk om het goed toe te passen. In dit artikel gebruik ik een aantal principes die onder andere ook terug te vinden zijn in adviezen van de originele paper van het NIST (National Institute of Standards and Technology) en adviezen van onder andere het NCSC (National Cyber Security Centre). In het "Zero Trust Architecture" paper wordt gesproken over 7 principes van zero trust, waarvan ik er in dit artikel een aantal wil bespreken namelijk:

> "All data sources and computing services are considered resources."
> "Access to resources is determined by dynamic policy—including the observable state of client identity, application/service, and the requesting asset—and may include other behavioral and environmental attributes"
> "All resource authentication and authorization are dynamic and strictly enforced before access is allowed"

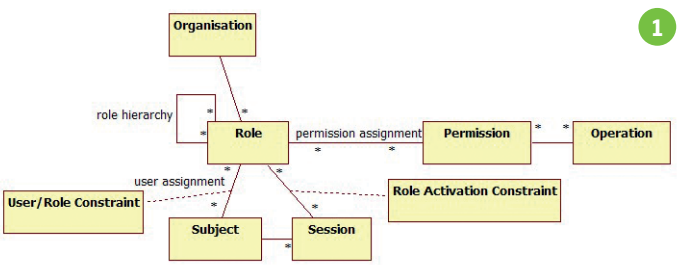
Deze principes zijn interessant voor veel developers omdat het iets zegt over hoe je toegang tussen applicaties, services en gebruikers moet authentiseren en autoriseren. Met het implementeren van deze 3 zero trust principles ben je er nog niet, maar het gaat voor dit artikel te ver om ook de andere principles toe te lichten.

RBAC-Model

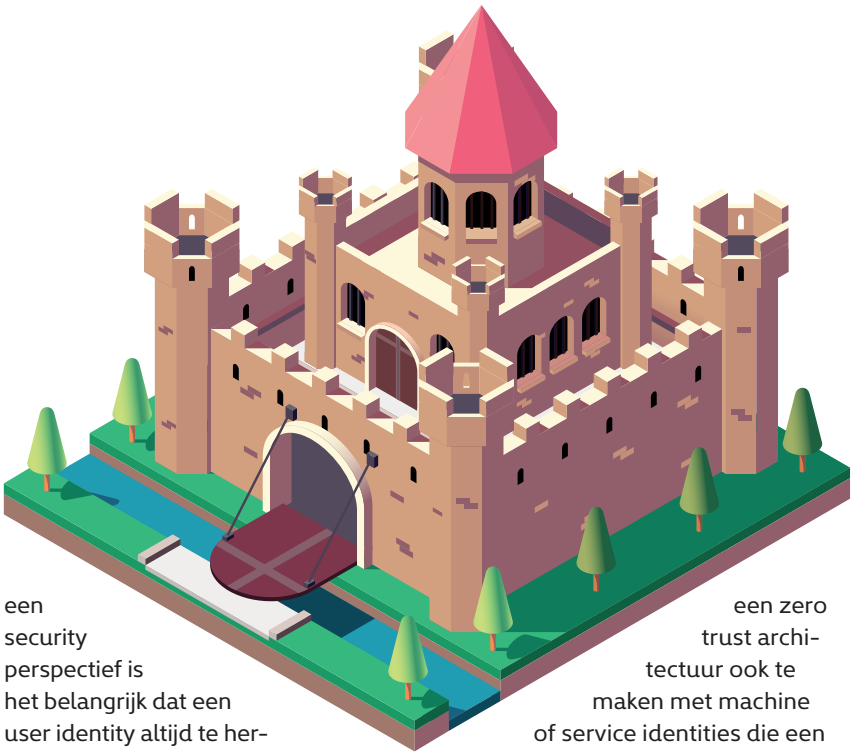
Role based access control is in veel gevallen een goed model om een zero trust architectuur mee te ontwikkelen. Het voorziet in elementen om zonder harde perimeters een zero trust architectuur te ontwerpen en te implementeren die toch veilig is. **Listing 1**.
Een RBAC-model kent doorgaans een subject (vaak ook wel identity) genoemd dat staat voor een gebruiker of andere agent in de architectuur. Dat kan dus bijvoorbeeld een ingelogde gebruiker, service of device zijn. Een subject is aan één of meerdere roles toegewezen welke weer één of meerdere permissies bevat. Een permissie geeft uiteindelijk toegang tot een bepaalde operatie of resource.
De API's, applicaties of services in de architectuur moeten valideren of een subject zich geauthentiseerd heeft bij de gedeelde identityserver aansluitend dient het subject de juiste permissies te bevatten om de operatie uit te kunnen voeren.

Identity Management

Het fundament van een goede RBAC-implementatie ligt in een single source van (subjects) identities. Steeds meer organisaties zien de interne gebruiker als een van de belangrijkste risico factoren. Dit komt omdat zij vaak fouten maken door bijvoorbeeld in phishing mails te trappen of doordat ze wachtwoorden hergebruiken tussen privé en werk accounts. Vanuit



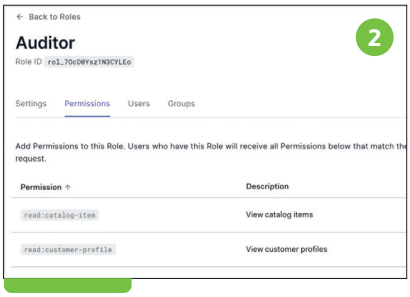
1



een security perspectief is het belangrijk dat een user identity altijd te herleiden is naar een daadwerkelijke gebruiker en idealiter dat hetzelfde account zoveel mogelijk gebruikt wordt in het IT-landschap (SSO). In de praktijk kan dit bijvoorbeeld het corporate account van een gebruiker zijn waarmee de gebruiker ook toegang krijgt tot de cloud omgevingen.
Naast user accounts hebben we in een zero trust architectuur ook te maken met machine of service identities die een rol spelen in het landschap. We kunnen er namelijk niet meer vanuit gaan dat alles wat achter de een bepaalde perimeter leeft veilig en een good actor is. Het principe van zero trust houdt in dat elke identiteit zowel users, services als machines zich kunnen authentiseren en autoriseren.
Op OpenID of OAuth gebaseerde identity services gebruiken doorgaans 1 of meerdere identity providers om de gebruiker te authentiseren, dit kan bijvoorbeeld een Azure Active Directory, Facebook of Google account zijn of eventueel een eigen ontwikkelde identity provider. In OpenId Connect kunnen clients op een bepaalde manier een gebruiker authentiseren bij de OpenId Connect server, dit noemen we de grant type. Voor server to server communicatie wordt hier vaak de client_credentials flow voor gebruikt waarbij een client_id en client_secret worden uitgegeven waarmee de service, applicatie of device zich kan authentiseren.
Ik zou aan raden om een bestaande dienst af te nemen als identity en access management service. Dit kan een 3rd Party SaaS dienst zijn zoals Auth0, Okta of Ping maar het kan ook een service zijn die beschikbaar is in Azure zoals Azure B2C of Azure Active Directory. Veel organisaties hebben in het verleden ook gebruik gemaakt van IdentityServer als OpenId Connect Server die zelf te hosten en uit te breiden is.

Autorisatie policies

Het is inmiddels vrij standaard om voor publieke API's een autorisatie model te implementeren. Dit doen we vaak door bepaalde operaties al dan niet beschikbaar te maken voor de ingelogde gebruiker op basis van de gegevens in bijvoorbeeld het JWT-token. Een van de zero trust principes schrijft echter voor dat het autorisatie model niet hard coded maar dynamisch moet zijn. Dynamisch is een vrij ambigu begrip maar wat het in ieder geval betekent is dat het autorisatie model niet in de code van individuele services/API's moet liggen. De vuistregel die ik vaak hanteer is dat het toekennen van rollen aan gebruikers of services configurabel moet zijn en zonder redeploy van services geeffectueerd moet kunnen worden.
Veel identity en access management-diensten zoals Auth0, hebben functionaliteit beschikbaar om rollen en permissies te definiëren. De makers van IdentityServer hebben hiervoor ook een product ontwikkelt genaamd PolicyServer. In alle gevallen is het mogelijk om een boom van policies inclusief rollen en permissies (autorisaties) vast te leggen. Om het iets concreter te maken heb ik 2 voorbeelden geselecteerd. **Listing 2**.



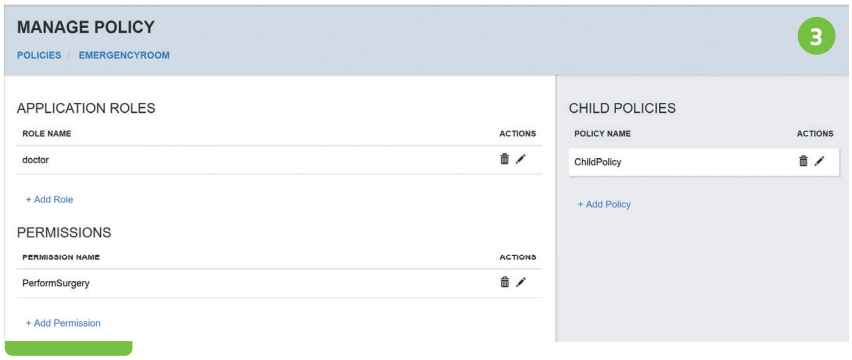
Voorbeeld Auth0

BIO

Dibran Milder

Dibran Mulder is een Cloud Solution Architect werkzaam bij Cloud Republic. Dibran is co-host van devtalks.nl een podcast voor software developers in Nederland.





Voorbeeld PolicyServer

Rollen in Auth0 worden gekoppeld aan permissies die vanuit de API's worden gedefinieerd. Een rol geeft dus effectief toegang tot een set van permissies welke in feite de rechten zijn voor toegang tot de specifieke API's. Gebruikers kunnen bepaalde rollen toegewezen krijgen. **Listing 3**.
PolicyServer van IdentityServer kent het concept Policies welke Rollen bevatten. Rollen bevatten weer Permissies. Het model verschilt iets met Auth0 doordat het een Policies entiteit bevat die weer child Policies onderkent. Een policy is eigenlijk een verzameling van rollen die voor een bepaalde applicatie of service benodigd zijn.

User role constraint

Het complexe gedeelte van een goede RBAC-implementatie zit hem vaak in de user/subject role constraint, in andere woorden: hoe zorg je ervoor dat een gebruiker of service de juiste rollen en rechten toebedeeld krijgt? Auth0 en PolicyServer hebben hierin een verschillende implementatie die op subdelen wel overeenkomt. Belangrijk is, is dat je de ingelogde identity kunt koppelen aan rollen. In Auth0 kan je dit doen door simpelweg de gebruiker te koppelen aan rollen of de applicatie te koppelen aan permissies. Je kunt dit ook meer dynamisch maken door "Auth Pipelines" te gebruiken en een rule te ontwikkelen in JavaScript die op basis van bijvoorbeeld Active Directory Groepen bepaalde rollen toekent aan een gebruiker. Het mooie aan Auth0 is dat het extensible is doormiddel van code snippets toe te voegen aan de authenticatie pipeline. **Listing 4**.
Bij PolicyServer ligt het iets anders, een identity maakt aanspraak op een rol als het voldoet aan 1 van de 3 opties, namelijk: User ID assignment, Identity Role assignment en Claims Evaluation assignment. Daarmee is het in PolicyServer mogelijk om individuele users rechten te geven op bepaalde rollen, maar je kunt ook een groep van users rechten geven door in het Authenticatie JWT token een IdentityRole mee te geven in de claim: Role. Je zult dan wel in de identity server oplossing ervoor moeten zorgen dat je gebruikers een role claim kunt geven op basis van bijvoorbeeld Azure Active Directory groepen of andere logica. De laatste optie is om een Claims Evaluation assignment toe te voegen waarin je simpelweg een C# expression schrijft waarmee je een custom validatie kunt uitvoeren. Zoals onderstaand voorbeeld beschrijft: **Listing 5**.

Evaluatie van Authenticatie en Autorisatie in API's

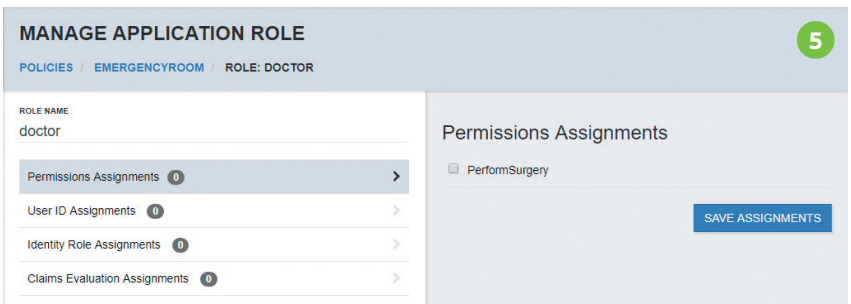
De laatste stap is uiteraard de evaluatie van de identity en de autorisatie van de API of service die wordt aangeroepen. Het grote verschil tussen Auth0 en PolicyServer zit hem in het feit dat bij Auth0 het JWT-token verrijkt is met claims die in het autorisatie model zijn geconfigureerd. In het geval van PolicyServer wordt runtime geëvalueerd bij de

```
function addSalesEmployeeFromAzureActiveDirectory(user, context, callback) {
    var salesGroup = 'SalesEmployees';
    if (user.groups) {
        if (typeof user.groups === 'string') {
            user.groups = [user.groups];
        }
        var userIsSalesEmployee = user.groups.some(function (group) {
            return salesGroup === group;
        });

        if (userIsSalesEmployee) {
            context.idToken['https://example.com/roles'] = ["SalesEmployee"];
        }
    }

    callback(null, user, context);
}
```

4




```
user => user.HasClaim(claim => claim.Type == "department" && claim.Value == "sales")
```

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddAuthentication(options =>
        {
            options.DefaultAuthenticateScheme = JwtBearerDefaults.Authenticati-
onScheme;
            options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
        }).AddJwtBearer(options =>
        {
            options.Authority = $"https://{Configuration["Auth0:Domain"]}/";
            options.Audience = Configuration["Auth0:Audience"];
        });

        services.AddAuthorization(options =>
        {
            options.AddPolicy("WriteAccess", policy =>
                                policy.RequireClaim("permissions", "create:invoice",
"update:invoice"));
            options.AddPolicy("DeleteAccess", policy =>
                                policy.RequireClaim("permissions", "delete:invoi-
ce"));
        });
    }
}
```

6

```
[ApiController]
[Route("api/[controller]")]
public class InvoiceController : ControllerBase
{
    [HttpPost]
    [Authorize(Policy = "WriteAccess")]
    public ActionResult PostInvoice(Invoice invoice)
    {
    }
}
```

7

```
public class Startup
{
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();

        services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
            .AddJwtBearer(options =>
            {
                // base-address of your identityserver
                options.Authority = "https://demo.identityserver.io";

                // name of the API resource
                options.Audience = "api1";
            });

        services.AddPolicyServerRuntimeClient(Configuration.GetSection("PolicySer-
verRuntimeClient"))
            .AddAuthorizationPermissionPolicies();
    }
}
```

8

```
{
    "PolicyServerRuntimeClient": {
        "PolicyServerUrl": "https://[policyserverdomain]",
        "BasePolicy": "BEHEER",
        "TokenClient": {
            "Authority": "https://[identityserverdomain]",
            "ClientId": "policy.runtimeclient"
        }
    }
}
```

9

```
Net zoals bij Auth0 kan nu in de controller of action afgedwongen worden dat de ge-
authentiseerde identity de juiste permissies heeft om de API te benaderen.
[ApiController]
[Route("api/[controller]")]
public class InvoiceController : ControllerBase
{
    [HttpPost]
    [Authorize("create_invoices")]
    public ActionResult PostInvoice(Invoice invoice)
    {
    }
}
```

PolicyServer. Beide oplossing hebben z'n eigen voor- en nadelen, zo is Auth0 waarschijnlijk iets performanter, maar duurt een aanpassing in het autorisatie model wel net zolang als de lifetime van het JWT token.

Voorbeeld implementatie van Auth0

Allereerst moet uiteraard de Auth0 authenticatie provider toegevoegd worden aan de ASP.net core middleware. Vervolgens wordt het ASP.net Autorisatie model van de API gelinkt aan het autorisatie model van Auth0 welke meekomt met specifieke claims in het JWT-token. **Listing 6**.

In de controller of action kan nu afgedwongen worden dat de geauthentiseerde identity de juiste permissies heeft om de API te benaderen. **Listing 7**.

Voorbeeld implementatie van PolicyServer

Net zoals bij Auth0 moet uiteraard eerst authenticatie provider toegevoegd worden aan de ASP.net core middleware. Vervolgens wordt het autorisatie model van ASP.net middels de Nuget package: "PolicyServer.Runtime.Client.AspNetCore" gekoppeld aan de PolicyServer. Daarbij wordt een Base Policy verwacht waartegen de ingelogde identity geëvalueerd zal worden. **Listing 8**.

De AddPolicyServerRuntimeClient verwacht een aantal appsettings, waaronder de base policy. **Listing 9**.

Samenvatting

In een zero trust architectuur authentiseren we elke subject of identity in het landschap. Daarbij hanteren we een autorisatie model die we doorgaans vastleggen bij onze centrale identityprovider of daar een product zoals PolicyServer voor aanschaffen of ontwikkelen. Op basis van de geauthentiseerde identiteit kan de API controleren of opvragen of de identiteit wel de juiste permissies heeft om de operatie uit te voeren. ●

Teams apps ontwikkelen met Dataverse integratie

Veel lezers zullen wel bekend zijn met een van de populaire front-end frameworks/libraries, zoals Angular, React, Vue.js, Blazor of een van de vele andere. De manier om hier snel professionele oplossingen in te ontwikkelen is ook flink verbeterd in de afgelopen jaren.

Auteur: Jan de Vries

De afgelopen tijd ben ik ook aan de slag geweest met het Power Platform, en dan met name Power Apps, Power Automate & de Dataverse integratie. Deze combinatie van oplossingen stelt ontwikkelaars, en mensen van 'de business' in staat om snel oplossingen te maken.

Power Platform? Dat is toch niet voor ontwikkelaars!

Het Power Platform van Microsoft staat bekend als een low-code/no-code platform en dat is het ook. Het staat zelfs op de landing pagina:

Omdat dit het enige platform is met een set tools met weinig code die je teams nodig hebben om flexibel te blijven en oplossingen te bouwen die je bedrijf vooruithelpen. Microsoft Power Platform is vanaf het begin opgebouwd met het oog op sterke beveiliging, beheer en naleving en biedt de ruimte om te groeien.
Bron: https://powerplatform.microsoft.com/nl-nl/what-is-power-platform/

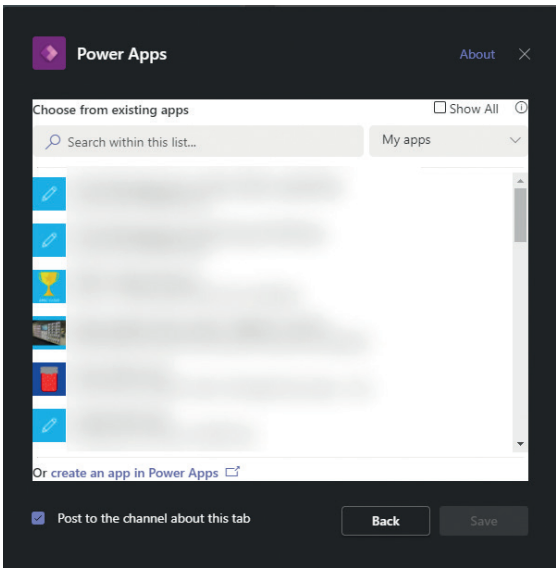
Dit klinkt niet heel aantrekkelijk voor een ontwikkelaar, want 'wij' vinden het vaak veel leuker om zelf een hele oplossing te maken, denk aan integraties, schermen, communicatie, componenten, beveiliging, etc. En in veel gevallen doen we dit dan ook 'beter' als in een low-code/no-code platform.
Het woord 'beter' staat tussen aanhalingstekens met een reden. Want hoewel we de oplossingen vaak wel 100% op maat kunnen maken voor een specifieke klant, betekent dit vaak wel veel ontwikkelwerk, zelf alles moeten testen, bugs oplossen, etc. Dit kan redelijk in de papieren lopen voor wat betreft tijd en geld.

Of de uiteindelijke oplossing dan nog steeds 'beter' is, in vergelijking met een oplossing die, bijvoorbeeld, in het Power Platform is gemaakt valt te bezien. Vaak kun je daarmee met relatief weinig moeite toch een nette oplossing maken welke goed genoeg voor het vraagstuk dat de business opgelost wil zien.
Het aantal connectoren, voor nagenoeg iedere populaire service, dat standaard is te gebruiken is ook immens. Dit betekent dat je niet meer zelf een integratie met bijvoorbeeld SAP, Dynamics of Azure Service Bus hoeft te ontwikkelen, maar dit out-of-the-box kunt gebruiken. De connectoren zijn goed te vergelijken met degene die ook in Azure Logic Apps gebruikt kunnen worden.
Mocht het zo zijn dat jouw connector voor een service er echt niet tussen staat, of je hebt echt maatwerk nodig, dan staat het vrij om zelf een connector te maken & publiceren in de publieke of private store.
Binnen het Power Platform wordt ook een database aangeboden, genaamd Dataverse. Dit kun je zien als SQL Server on steroids. In de basis draait het op een SQL Server, maar er zijn veel toevoegingen op gemaakt, waardoor het ideaal geschikt is om gebruikt te worden binnen het Power Platform. Dataverse is volledig gemanaged, is relatief eenvoudig te beveiligen, role-based-access-control is ingebouwd, en schaling wordt automatisch geregeld.

Microsoft Power Platform is vanaf het begin opgebouwd met het oog op sterke beveiliging, beheer en naleving en biedt de ruimte om te groeien.

Er is zelfs een speciale Dataverse versie voor Teams gemaakt:

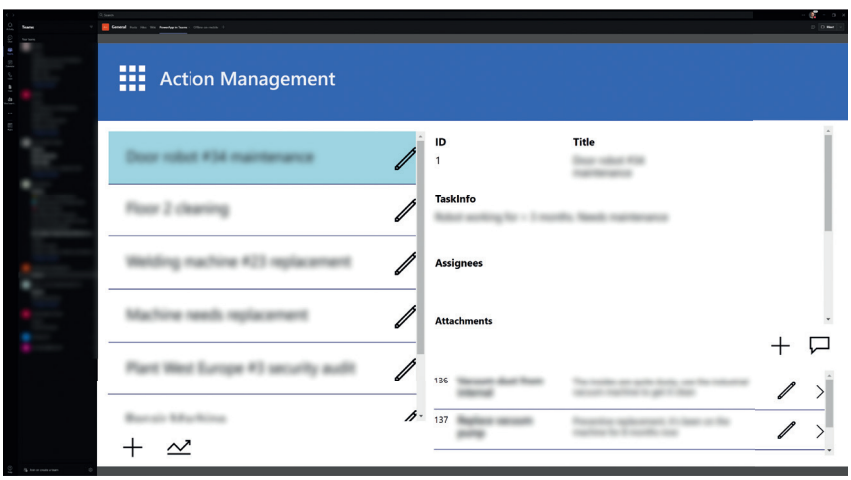
Make working in Microsoft Teams more productive and collaborative with Dataverse for Teams—a low-code data platform built into Teams. Remove friction for users having to switch between multiple services and apps with an end-to-end experience. Create secure, integrated solutions with Microsoft Power Apps and support your workforce with intelligent bots using Microsoft Power Virtual Agents.
Bron: https://powerplatform.microsoft.com/en-us/dataverse/



Mocht jouw organisatie dus geen licentie hebben om gebruik te maken van het Power Platform, dan is dit een (gratis) optie om toch aan de slag te gaan.
Natuurlijk zullen niet alle wensen van klanten, of de business, kunnen worden ingewilligd met standaard componenten. In het geval dat er een stuk functionaliteit ontbreekt en per se moet worden geïntegreerd in een applicatie, dan zijn er verschillende mogelijkheden om dit op te lossen. Een daarvan is het ontwikkelen van een PCF-component (https://docs.microsoft.com/en-us/power-apps/developer/component-framework/overview). Hiermee worden ontwikkelaars in staat gesteld om custom code componenten te maken, gemaakt met HTML, CSS & TypeScript. Het gebruik maken van dit type componenten is ideaal in het geval dat er kleine stukken functionaliteit moet worden toegevoegd aan een Power App.
Het maken van dit soort componenten wordt niet verder in dit artikel behandeld, maar mocht je hier interesse in hebben, dan kan ik van harte de documentatie pagina's die hierover gaan aanbevelen.
Veel applicaties die in het Power Platform worden ontwikkeld kunnen door zogenaamde 'citizen developers' worden gemaakt, dat is dan ook de kracht dat iedere medewerker binnen een organisatie iets voor zichzelf, of een team, in het werkproces kan automatiseren of vereenvoudigen.
Naar mijn mening is het wel zo dat het fijn is dat een ontwikkelaar betrokken is bij de ontwikkeling van (belangrijke) applicaties, om zo een beter te onderhouden & testen applicatie neer te zetten. Een ontwikkelaar heeft vaak meer ervaring met zaken als beveiliging, performance, onderhoudbaarheid en versiebeheer. Als dit goed wordt opgepakt, dan zitten er grote voordelen aan het gebruik van Power Apps (en het Power Platform in z'n algemeenheid) ten opzichte van op maat ontwikkelde applicaties/platformen.

Power Apps in Teams

Mocht je het nog niet hebben gedaan, dan zou ik aanraden om een 'Hello World'-achtige applicatie te maken met Power Apps en een integratie met Dataverse. Dit kan zowel met de volledige- als wel de Teams-variant. Bonuspunten voor het toevoegen van een Power Automate flow!
Zodra de Power App is ontwikkeld kan die worden toegevoegd als een nieuwe tab in Teams. Door de optie Power Apps te selecteren zal er



```
With(
{
    tmpGroupID: Param("groupId"),
    tmpChannelId: Param("channelId"),
    tmpTeamId: Param("teamId")
},
    If (
        !IsBlank(tmpGroupID),
        Set(
            appConfig,
            {
                TeamsData: {
                    GroupId: tmpGroupID,
                    ChannelId: tmpChannelId,
                    TeamId: tmpTeamId
                }
            }
        )
    )
);
```

1

een pop-up verschijnen waar alle gepubliceerde Power Apps kunnen worden geselecteerd binnen de organisatie.
Een veelvoorkomend voorbeeld dat men vaak maakt is een To Do applicatie, zo heb ik ook een 'Action Management' applicatie ontwikkeld met daarin verschillende Power Automate flows en een geïntegreerd Power BI dashboard.
De basis voor deze applicatie kan binnen enkele uren worden opgezet en daarmee kunnen gebruikers direct aan de slag. Zeker met de verschillende Power Automate flows is het mogelijk om integraties met externe systemen toe te voegen zonder daar een expert voor hoeven te zijn.
Natuurlijk kun je deze Power App ook prima los starten, maar het grote voordeel om dit binnen Teams te doen, is dat de applicatie ook gebruik kan maken van de context waar die in draait (welk team, deelnemers in het team, kanalen, etc.). Deze informatie wordt in de querystring van de Teams-tab meegestuurd en kan eenvoudig worden gebruikt in de applicatie in bijvoorbeeld een `appConfig` variabele. **Listing 1.**
Een ander bijkomend voordeel, vanuit een gebruiker perspectief, is dat deze persoon minder vaak hoeft te switchen tussen applicaties, wat de productiviteit ten goede kan komen. Het namelijk mogelijk om vanuit de Power App een chat te starten met collega's en gelijktijdig door te werken.

Power Apps in Teams, doen of niet?
Mijn korte antwoord is: JA!
Maar zoals iedereen weet zijn er geen absolute antwoorden in onze wereld. Het antwoord zal sterk afhangen van de use-case en wensen van de gebruikers/business.
Wel is het zo dat het relatief eenvoudig is om een, eenvoudige, applicatie op te zetten. De applicaties kunnen ook prima met verschillende APIs communiceren en desgewenst kan er voor deze APIs met enkele klikken een Custom Connector worden gemaakt. Op deze manier hoeven ontwikkelaars minder bezig te zijn met het maken van de zoveelste CRUD-applicatie in hun carrière en bezig te houden met interessantere vraagstukken, zoals schaalbaarheid, beveiliging, performance, etc.

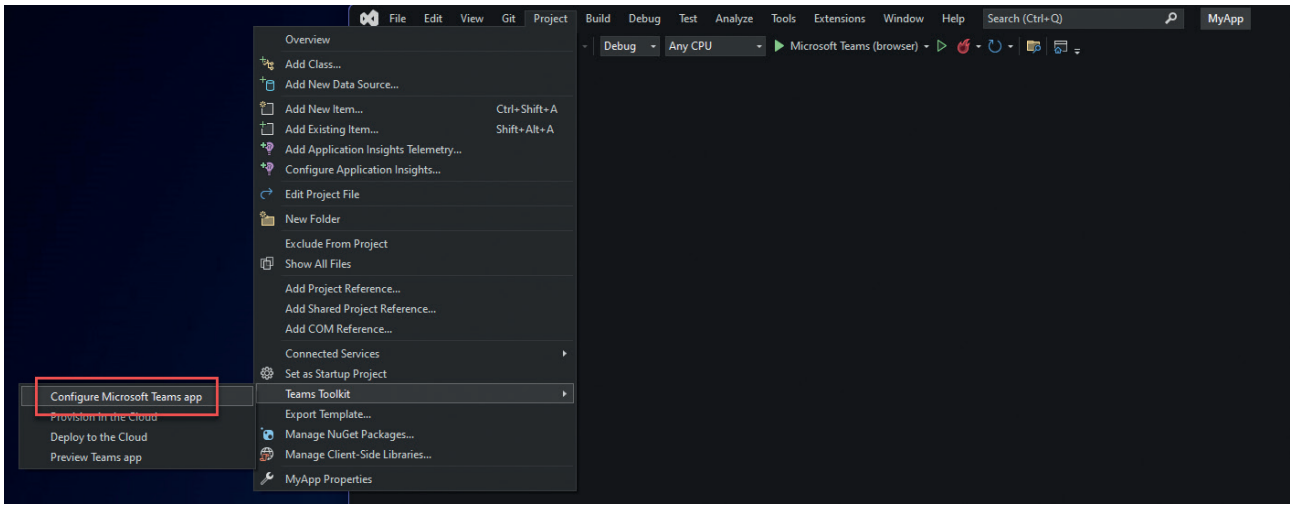
Ik wil of moet nog steeds custom oplossingen ontwikkelen!
Dat begrijp ik!
In de afgelopen jaren dat ik bij verschillende klanten projecten heb mogen uitvoeren heb ik geleerd dat er geen een klant gelijk is en ieder bedrijf vaak wel iets 'raars' in het werkproces heeft dat je nog niet eerder bent tegengekomen. Dit is niet erg en maakt ons werk, naar mijn mening, des te meer interessant!
Vaak is het wel dat dit 'ene' unieke stuk in het werkproces vaak maar een klein onderdeel is van het grotere geheel. Daarom denk ik dat je vraagstukken van de business wel voor 80%, of misschien wel 90%, de applicaties met een Power Apps kunt oplossen, al dan niet met kleine PCF-componenten.
Voor de resterende functionaliteit kan dan nog een zelf ontwikkelde applicatie worden gemaakt. Hier kan ervoor worden gekozen om een applicatie te ontwikkelen met Angular, React, Vue.js, etc., maar er is ook een minder bekende optie, namelijk een Native Teams applicatie. Dit kan een hele logische keuze zijn, aangezien alle andere applicaties (Power Apps) ook binnen Teams kunnen worden gehost. Is dit heel anders als het hosten van een webapplicatie in een `Web`-tab binnen Teams? Nee, niet heel anders, maar het heeft wel enkele voordelen zoals de integratie met Azure Active Directory & M365, de applicatie als een 'package' kunt aanbieden en er standaard gebruik kan worden gemaakt van de Fluent UI (https://docs.microsoft.com/en-us/fluent-ui/web-components/) welke ook door Teams zelf wordt gebruikt. Standaard wordt er ook gebruik gemaakt van Blazor, dus is het relatief eenvoudig aan te leren voor .NET ontwikkelaars.

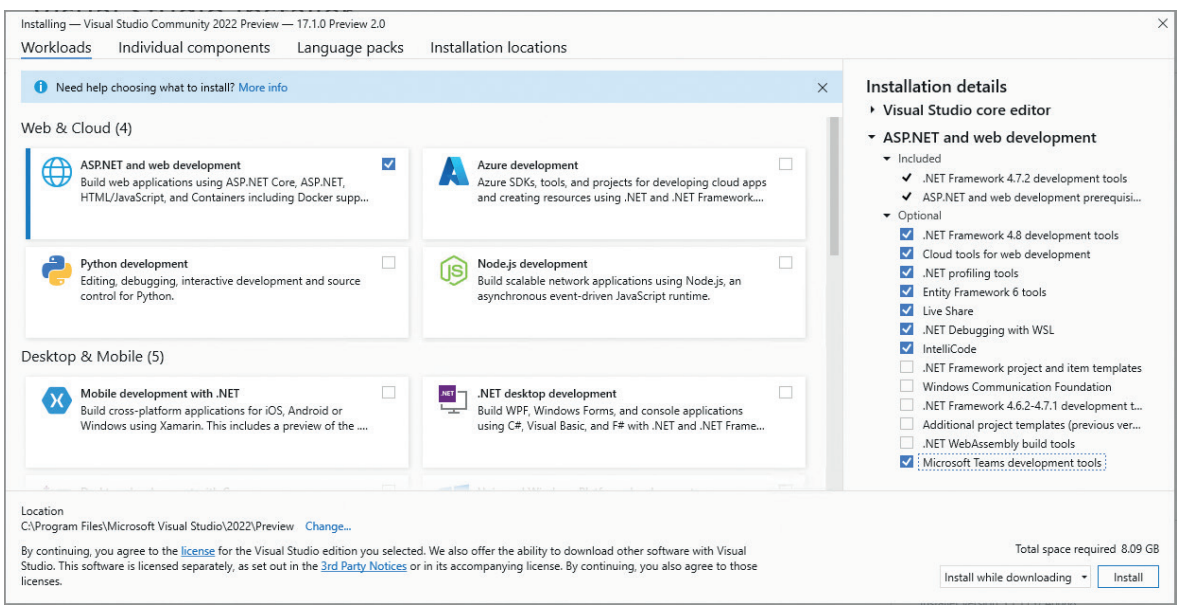
De manier om hier snel professionele oplossingen in te ontwikkelen is ook flink verbeterd in de afgelopen jaren.

Een native Teams app maken

Sinds Visual Studio 2022 is er de optie om de Microsoft Teams development tools te installeren. Het is aan te raden deze optie te gebruiken, want het installeert ook de Teams App project template.
Deze template geeft als optie om een Teams Application te maken met zowel `Tab`- als `Bot`-functionaliteiten. Het maken van een Bot is iets dat niet in dit artikel wordt behandeld, maar is zeker interessant!
De 'Get started quickly...' in de documentatie (https://docs.microsoft.com/en-us/microsoftteams/platform/toolkit/visual-studio-overview) geeft een goed overzicht wat er voor een snelle start gedaan moet worden.
Het belangrijkste vind ik dat de Teams applicatie goed wordt geconfigureerd in Azure & M365. Hier kan de optie voor worden gebruikt `Teams Toolkit -> Configure Microsoft Teams app`.
Deze optie zorgt er onder andere voor dat er een App Registration wordt aangemaakt in Azure Active Directory en de benodigde properties in het bestand `appsettings.Development.json` van het project worden gevuld met het `ClientId`, `ClientSecret` en `OAuthAuthority`.

De App Registration in Azure Active Directory gedraagt zich verder hetzelfde als een reguliere, dus ook hier is het mogelijk om API Permissions aan te vragen voor de applicatie, applicatie rollen toe te voegen, etc.
Wanneer er een native Teams applicatie wordt gemaakt is er een grote kans dat er ook al Power Apps zijn ontwikkeld binnen de organisatie. Deze Power Apps maken waarschijnlijk ook gebruik van Dataverse als repository. Het ligt voor de hand dat de native Teams applicatie ook toegang moet krijgen tot deze data. Standaard is dit niet geregeld en moet deze toegang nog worden toegevoegd.





Het toevoegen van deze toegang kan worden gedaan op de `API permissions`-blade van de zojuist gegenereerde App Registration. Hier moet de toegang tot `Dynamics CRM` worden toegevoegd, zoals is te zien in de afbeelding.
Let op dat de `Grant admin consent for [organization]` wel moet worden verleend. Mocht dit niet worden gedaan, dan zullen de gebruikers een fout te zien krijgen bij het verlenen van hun 'consent' aan deze applicatie.
Niet iedereen binnen de organisatie heeft voldoende rechten om deze 'admin consent' te activeren, dus let daarop!

Data ophalen in de nieuwe native Teams applicatie

Nadat de applicatie volledig is geconfigureerd is het eindelijk tijd om te ontwikkelen!
Het standaard project template heeft een voorbeeld applicatie welke in staat is om data over de ingelogde gebruiker op te halen vanuit Azure Active Directory. Dit is mogelijk, omdat de gebruiker al in Teams is ingelogd en de applicatie de `User.Read` autorisatie heeft op de Microsoft Graph.
Omdat er ook data uit Dataverse moet worden verkregen, of toegevoegd, moet er door de gebruiker ook toegang worden verkregen tot deze bron. Het is hiervoor noodzakelijk om het Web API endpoint van de betreffende omgeving te verkrijgen. Deze kan in Power Platform worden gevonden onder `Settings -> Developer Resources`.
Om te kunnen lezen en schrijven kan het eigen account van de gebruiker worden gebruikt, want het is in Dataverse heel goed mogelijk om rechten toe te kennen op tabellen en kolommen.
In de applicatie dienen we nu alleen de Web API endpoint te definiëren en de scope (`user_imporsonation`). **Listing 2.**
In de Teams applicatie kan er de `TeamsUserCredential` worden geïnjecteerd. **Listing 3.**
Er kan nu een `LoginAsync` worden uitgevoerd met de benodigde scopes, waardoor de gebruiker consent kan geven om de Teams applicatie voor de geconfigureerde resources. **Listing 4.**
Zodra het consent is verleend is het mogelijk om de gebruikersgegevens uit de AAD te lezen, maar ook de

```
private string dynamicsResource = "https://[thePowerPlatformEnvironment].api.crm4.
dynamics.com/";
private string dynamicsScope = "user_impersonation";
```

2

```
@inject TeamsUserCredential teamsUserCredential
```

3

```
await teamsUserCredential.LoginAsync(new[] { _scope, dynamicsResource + dynamicsScope });
```

4

```
var dynamicsToken = (await teamsUserCredential.GetTokenAsync(new TokenRequestContex-
t(new string[] { dynamicsResource + dynamicsScope}), new System.Threading.Cancella-
tionToken())).Token;
var httpClient = HttpClientFactory.CreateClient();
httpClient.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.Authen-
ticationHeaderValue("Bearer", dynamicsToken);
var whoAmIResponse = await httpClient.GetAsync(dynamicsResource + "/api/data/v9.2/
WhoAmI");

if (whoAmIResponse.IsSuccessStatusCode)
{
    JObject body = JObject.Parse(whoAmIResponse.Content.ReadAsStringAsync().Result);
    Guid userId = (Guid)body["UserId"];

    _dynamicsUserId = userId.ToString("D");
}
else
{
    ErrorMessage += "Web API call failed";
    ErrorMessage += "Reason: " + whoAmIResponse.ReasonPhrase;
}
```

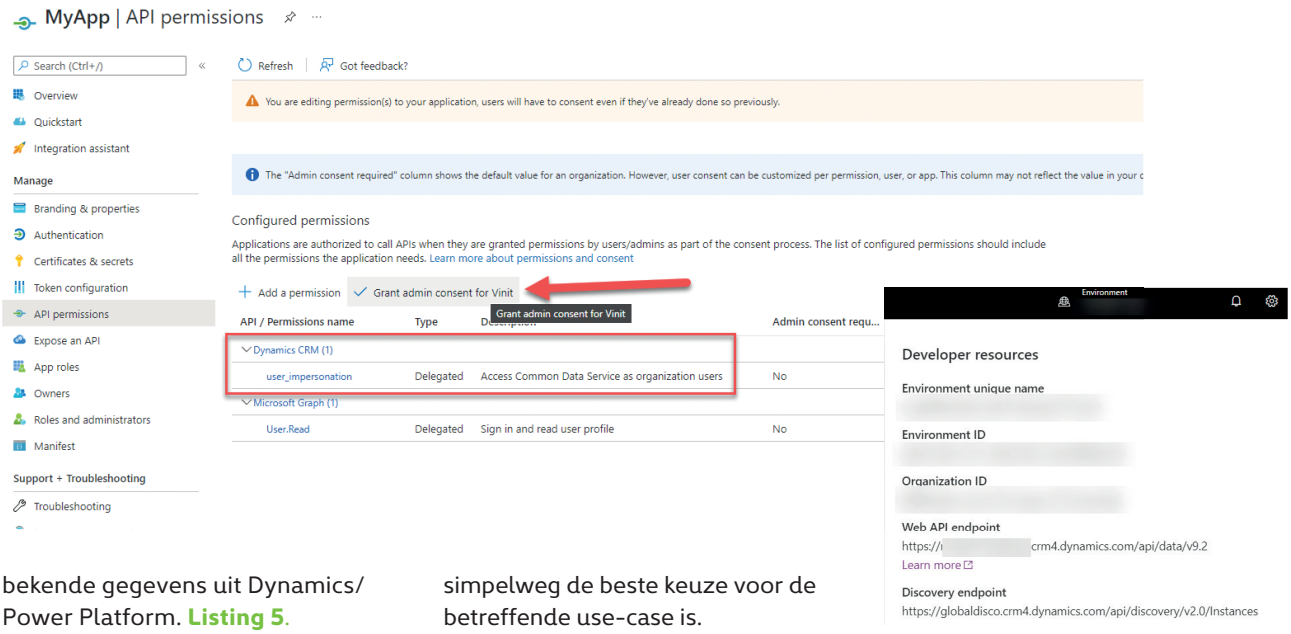
5

```
var dynamicsToken = (await teamsUserCredential.GetTokenAsync(new TokenRequestContex-
t(new string[] { dynamicsResource + dynamicsScope}), new System.Threading.Cancella-
tionToken())).Token;
var httpClient = HttpClientFactory.CreateClient();
httpClient.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.Authen-
ticationHeaderValue("Bearer", dynamicsToken);
var accountsResponse = await httpClient.GetAsync(dynamicsResource + "/api/data/v9.2/
accounts?$select=name,revenue&$orderby=revenue desc&$top=3");

if (accountsResponse.IsSuccessStatusCode)
{
    var topAccountsWithRevenue = await JsonSerializer.DeserializeAsync<Models.Data-
verse.Accounts.WithRevenue.Rootobject>(
                                    await accountsResponse.Content.ReadAs-
StreamAsync());
    _accountCollection.Clear();
    _accountCollection.AddRange(topAccountsWithRevenue.value);
}
else
{
    ErrorMessage += "Web API call failed";
    ErrorMessage += "Reason: " + accountsResponse.ReasonPhrase;
}
```

6

Het snel kunnen ontwikkelen, beschikbaar maken en valideren van een oplossing bij eindgebruikers heeft veel voordeel.



bekende gegevens uit Dynamics/ Power Platform. **Listing 5**.
Het aanroepen van het `WhoAmI`-endpoint is goed om te valideren dat de connectie goed is opgezet. Een interessantere call is om bijvoorbeeld te zoeken naar de top 3 accounts van het bedrijf, gesorteerd op omzet. Ook dit kan met een vergelijkbare call worden verkregen. **Listing 6**.
Redelijk veel code, en te veel om over te typen. De volledige oplossing is dan ook beschikbaar op GitHub: https://github.com/Jandev/TeamsAppWithDataverseApiConnection

Heeft dit allemaal nou veel voordelen?
Ik vind van wel.
Het snel kunnen ontwikkelen, beschikbaar maken en valideren van een oplossing bij eindgebruikers heeft veel voordeel. Zo kan er sneller worden geschakeld en gevalideerd of de gemaakte oplossing(en) voldoen. Mocht het nodig zijn om toch een stukje maatwerk (component of Teams applicatie) te ontwikkelen, dan behoort dat ook tot de mogelijkheden, maar zal eerder een uitzondering zijn. En het staat natuurlijk nog steeds vrij om een volledig losstaande applicatie naast de overige applicaties te ontwikkelen, omdat dat simpelweg de beste keuze voor de betreffende use-case is.
Het is alleen goed om te weten wat de (on)mogelijkheden binnen het Power Platform zijn, zeker met de aanwezige Teams integratie. Het zorgt ervoor dat wij ontwikkelaars ons minder druk hoeven te maken met 'saaie' CRUD-schermen en de focus meer gelegd kan worden op het verbeteren en automatiseren van processen en de cross-cutting concerns. ●

BIO

Jan de Vries

De afgelopen jaren heeft Jan als consultant vele rollen gehad. Sinds de opkomst van Azure heeft hij zich daar volledig op gestort. Zijn focus ligt dan ook bij het opzetten van schaalbare en performante software-oplossingen, waar Microsoft Azure een primaire rol in speelt.
Hij heeft een sterke voorkeur voor het gebruik van PaaS oplossingen.
Tegenwoordig werkt hij als Software Engineer bij Microsoft in een Azure product team, dus zit hij nog dichter bij de actie als voorheen! Jan blogt met enige regelmaat op https://jan-v.nl en kunt hem uiteraard ook op Twitter vinden met de handle @Jan_de_V





In zijn ene hand een computermuis en aan zijn andere hand een schaap en een koe. Dat beeld krijg je niet meer van je netvlies na een gesprek met Kornelis Hietkamp. Als software developer, eigenaar van een boerenbedrijf in Kollumerzwaag én trotse vader van de driejarige Auke heeft Kornelis volle weken. Daarnaast is hij in zijn spaarzame vrije tijd óók nog eventjes weidevogelbeheerder. Wat jij je als lezer nu waarschijnlijk afvraagt, vragen wij hem: hoe dan?!
Zorg, ICT of de natuur? Kornelis kan nauwelijks kiezen. En waarom zou hij ook? Hij kan zijn drie passies prima combineren. Aan de rand van natuurgebied Zwagermieden staat het familiebedrijf dat hij samen met zijn vrouw, zus en zwager runt. Daar houden zij zo'n honderdtwintig Texelaar-schapen, die elk jaar 1,5 gemiddeld lammetje krijgen. "De Texelaar-schaap is de Tesla onder de schapen", vertelt de multitasker trots. Daarnaast heeft de familie Hietkamp ongeveer vijfendertig koeien, die elk jaar een kalf krijgen, en een aantal varkens en paarden.

Werken in de Zorg-ICT

Bij het binnenwandelen van de schuur vertelt Kornelis hoe hij in de Zorg-ICT belandde. "Ik studeerde Informatica aan de Hogeschool van Leeuwarden en begon in 2010 als afstudeerder bij ChipSoft. Naast werken op de boerderij zocht ik een manier om mijn passie voor ICT op een nuttige manier in te zetten. De Zorg-ICT leek mij hiervoor de ideale sector, omdat ik altijd graag mensen help."
Zijn afstudeerstage resulteerde in een vast dienstverband, met een werkverdeling van 32 uur bij ChipSoft én 32 uur op de boerderij. Druk, maar vooral leuk, vindt Kornelis. "Als software developer werkte ik in eerste instantie vanuit ons toenmalige kantoor in Drachten. Later verhuisden we naar Heerenveen. De eerste jaren werkte ik aan software voor de radiologieafdeling, wat nu wordt doorontwikkeld door onze ontwikkelaars in Hoogeveen. Na een jaar van overdragen ben ik zelf weer terug in het kantoor van Heerenveen."

Efficiënte zorg

De tijd dat Kornelis werkte aan de radiologiesoftware droeg hij ook bij aan de bredere beschikbaarheid van die software. "Daardoor is het nu ook inzetbaar voor andere functieafdelingen, zoals de cardiologie-, long-, kaak- en KNO-afdelingen. Daarmee zorg ik dat zij hetzelfde, gestandaardiseerde proces krijgen rondom radiologie als de radiologieafdeling zelf. Die uniformiteit leidt tot meer duidelijkheid en efficiënte zorg."
In Heerenveen werkt Kornelis verder aan software voor de kliniek. "Mijn verhuizing leidde dus niet zozeer tot een functieverandering, maar tot een pakketverandering. Erg leuk, want daardoor word je weer op scherp gesteld." De softwareboer stak meteen veel tijd in het leren kennen van de software die Team Kliniek schrijft. "Processen voor de kliniek zijn namelijk anders dan voor de radiologie. Het is een mooi leerproces om je die specifieke code eigen te maken."
De energie die Kornelis in zijn eigen ontwikkeling steekt, komt de zorg ten goede. "Ik vind het belangrijk dat ik het zorgmedewerkers in Nederland zo gemakkelijk mogelijk maak om hun werk goed te doen. We willen toch allemaal goede zorg ontvangen als we die nodig hebben?! Dat heb ik zelf ervaren in mijn directe omgeving. Met de software die ik schrijf wil ik afdwingen dat zorgprofessionals alle mogelijkheden van HiX optimaal benutten, conform alle geldende wetten en regels, zodat patiënten op het juiste moment de juiste zorg krijgen, met een zo laag mogelijke registratielast voor de zorgprofessionals."
Die oplossingen programmeert Kornelis in C#. Voor de schermen van HiX gebruikt hij WPF-layout. Andere, veelgebruikte instrumenten zijn voornamelijk Microsoft-tools, waaronder Visual Studio. "Het is echt leuk om de nieuwste technieken in te zetten voor het oplossen van technische vraagstukken", vertelt Kornelis. "Dat ik dat kan doen voor een mooie sector als de zorg, maakt het helemaal mooi."
Kornelis zit helemaal op zijn plek. "Je hebt hier echt het gevoel dat je met elkaar en met klanten samenwerkt aan betere zorg." En dat valt prima te combineren met zijn andere passie, het boerenleven. "Gelukkig wel! Maar dat komt natuurlijk óók omdat ik mijn schapen heb geleerd niet tijdens kantoortijden te lammeren", besluit Kornelis met een knipoog. ●

> "Zorg-ICT is de ideale sector, omdat ik graag mensen help"

Impact maken in de Zorg-ICT?



werkenbijchipsoft.nl

ChipSoft



5 redenen waarom je blockchain development wilt leren als frontend developer

Wist je dat je met frontend skills kan bijdragen aan Web3 en de Metaverse? Misschien denk je: "maar dat heeft toch met blockchain te maken"? Zeker! Blockchain is een van dé bouwstenen van Web3 en de Metaverse. Web3 is een idee van de (opkomende!) derde versie van het internet waarbij blockchain wordt gebruikt voor de data van users. De Metaverse is een alternatieve virtuele wereld waarbij in combinatie met Web3 mensen veilig kunnen connecten, spelen en werken in deze virtuele werelden door middel van blockchain.

Auteur: Jessy The

Om hiermee te werken, moet je blockchain development kunnen. Je zult dus nieuwe dingen moeten leren maar het goede nieuws is dat je als frontender al 1-0 voor staat als je een zogenaamde "blockchain software developer" wilt worden.
Wat is blockchain development en hoe werken developers hierin? Let's dive in.
Blockchain developers bestaan doorgaans uit twee categorieën: blockchain software developers en blockchain core developers. De eerste groep zijn developers die nadenken over hoe een app eruit moet zien (yeah frontend!) én hoe de app werkt met de blockchain, bijv. door middel van smart contracts. De "core developers", houden zich o.a. bezig met het bouwen van protocollen, smart contract platforms en nodes voor de blockchain. For the record, jij behoort tot de eerste groep!
Bedenk eens hoe tof het zou zijn om naast het maken van frontend, je ook bezig te houden met blockchain development? Daarom ga ik in dit artikel in op 5 redenen waarom jij blockchain development zou moeten leren als frontender. Het is daarbij handig om eerst duidelijk te hebben wat blockchain, Web3 en de Metaverse inhouden en hoe zij zich contextueel tot elkaar verhouden.



Blockchain.Web3.Metaverse

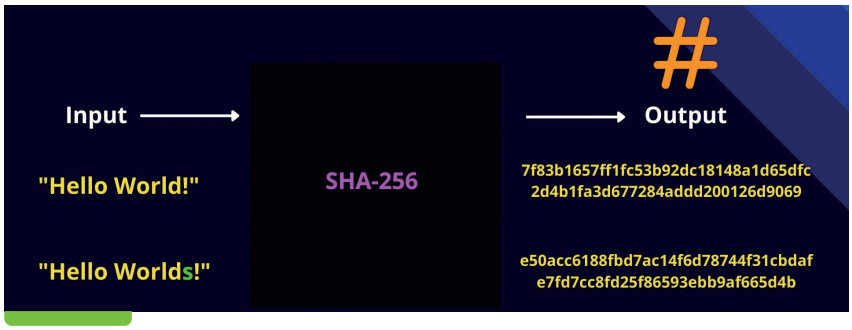
Ik ga hier zodanig diep op in, dat jij het zonder slag of stoot straks aan anderen kan uitleggen :).
Blockchain 101: De context van blockchain, Web3 en de Metaverse

Blockchain: a chain of blocks

Simpel gezegd: het is een chain of blocks. Elk block bevat data en wat dit inhoudt is afhankelijk van wat de blockchain ondersteunt. Het klassieke voorbeeld is Bitcoin en gezien dit een cryptocurrency is, worden in de blocks daarvan lijsten van transacties opgeslagen. Daar staat bijv. in dat Ernie gisteren tien euro aan Bert heeft betaald om 13:00 voor lunch. Daarnaast bevatten de blocks o.a. ook hashes die functioneren als unieke ID's van blocks. Een hash wordt gecreëerd wanneer een block wordt gemaakt en dit gebeurt meestal aan de hand van een hashing function genaamd SHA-256. Dit is een algoritme dat ongeacht de input, altijd een hash output bestaande uit symbolen en getallen gelijk aan 256 bits. De input in deze context is de data van het block, kortom: de transacties. Belangrijk om te onthouden is dat als je de input maar iets verandert, de output (de hash) dan totaal verandert.
De hash is tevens hetgeen waardoor blockchain een chain-constructie heeft zoals een linked list. Elk block bevat namelijk ook de hash van het block voorafgaand in de chain.
Stel: je bent Bert (én een hacker) en je vindt dat Ernie jou gisteren eigenlijk 500 euro had moeten betalen voor de lunch en je verandert deze transactie in het block.
Wanneer Bert de transactie verandert, verandert de hash van dit block met als gevolg dat het opvolgende block niet meer de juiste hash van dat block heeft. BAM! De chain breekt. Bert kan niet zomaar ongezien stiekem iets aanpassen op de blockchain. Alle data die op de blockchain staat is daarmee onveranderbaar en dit is één van de superpowers van blockchain.



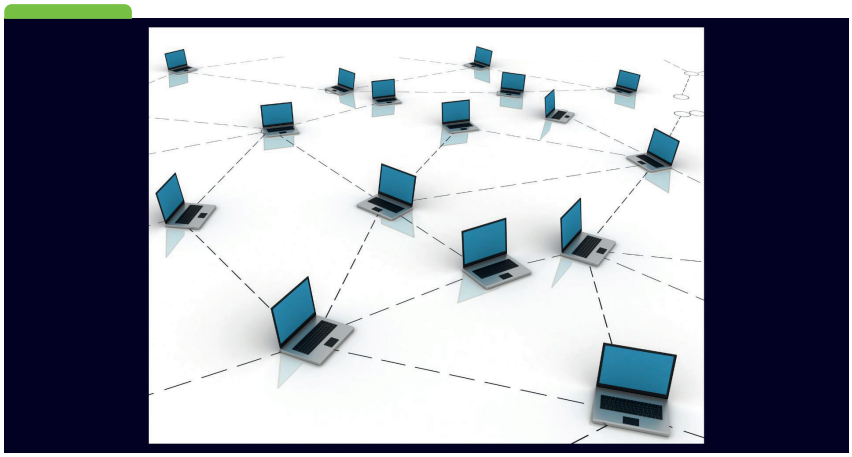
ErnieBertTransactie

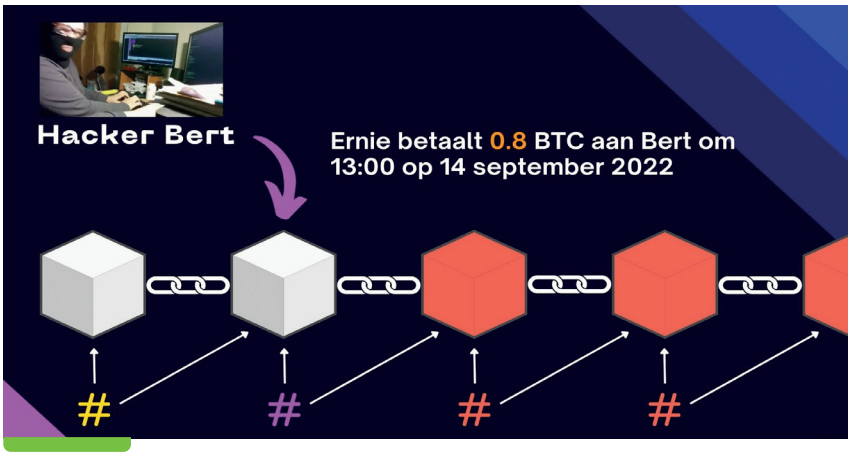


HashSHA256

Daarnaast wordt alles op de blockchain in realtime bijgehouden en bevatten alle blocks digitale timestamps. Het idee van het zetten van digitale timestamps in combinatie met de chain-constructie, behoren tot het gedachtegoed van Satoshi Nakamoto, de oprichter van Bitcoin. Hij wilde dat blockchain-technologie bijdraagt aan veiligheid, betrouwbaarheid, legitimiteit en transparantie. Een ander kenmerk van de blockchain wat hieraan bijdraagt, is dat het een distributed decentralised peer-to-peer network is. Je moet het zien als een netwerk van computers (nodes) die met elkaar communiceren zonder centrale organisatie. Op iedere node staat een copy van de gehele blockchain dus iedere node weet wat er op de blockchain gebeurt.
In de context van Bitcoin is een block vol bij ongeveer 1500 transacties.

DistribuedDecP2P





BertBreektChain

Wanneer het block vol is, vindt mining plaats waarbij het block aan de chain wordt toegevoegd. Herinner je je nog dat de hash een output is van de SHA-256? De output functioneert dus zowel als uniek ID als een wachtwoord. Miners proberen dit wachtwoord te kraken door de input voor de SHA-256 te raden aan de hand van allerlei berekeningen. Als een miner alle transacties van een block valideert en het password kraakt, moet het block nog worden geverifieerd door de overige nodes in het netwerk. Dit gebeurt aan de hand van consensus algorithms.
Een bekend consensus-algoritme hiervoor is die van Proof-of-work, origineel voorgesteld door Satoshi Nakomoto in zijn Bitcoin whitepaper[0].

Er moet "bewijs van het werk" zijn dat de miner heeft verricht. Een indicatie daarbij is o.a. de tijd die de miner nam om een block aan de chain toe te voegen. Hoe moeilijker het password is, des te langer een miner erover doet om dit voor elkaar te krijgen. Bij Bitcoin is de moeilijkheidsgraad van een password zodanig hoog, dat er minstens tien minuten zit tussen nieuw gevoegde blocks op de chain. Back to Bert, om de veiligheid, transparantie en betrouwbaarheid van de blockchain te laten zien. Bert heeft zich namelijk omgeschoold tot Super Hacker. In een poging om de chain niet te verbreken met zijn aanpassen, wilt hij een script schrijven om de hashes van alle opeenvolgende blocks snel te kunnen bijwerken. Mooi niet Bert! Dit is precies wat niet kan omdat het (bij Bitcoin) tien minuten duurt voordat een nieuw block aan de chain mag worden toegevoegd. Bert moet dan een kwantum computer bezitten met 1.9 biljoen qubits[1] om de blockchain van Bitcoin te hacken.

Nu je precies weet waar de krachten van blockchain liggen in veiligheid, betrouwbaarheid, transparantie en onveranderlijkheid, wordt het tijd om te kijken wat voor impact dit heeft op Web3 en de Metaverse.

Web3

Om alles gelijk helder te hebben: Web3 is niet gelijk aan Web3.0! Al gaat het hier wel om het idee van de derde versie van het internet. Web3.0 is Tim Berners Lee's visie over een semantic web. Web3 gaat over een gedecentraliseerd en blockchain-gebaseerd web en deze term is als eerste ooit genoemd door Gavin Wood, oprichter van Ethereum.

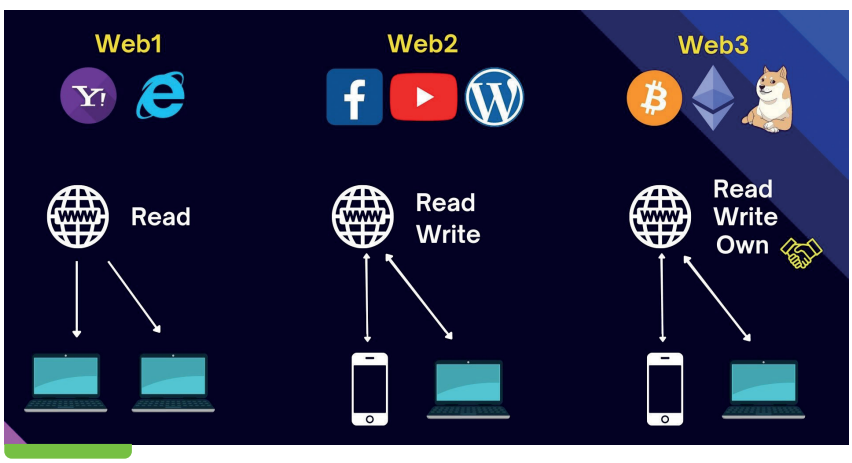
Laten we de historie van het web bekijken, aangezien Web3 de nieuwe versie ervan gaat zijn. Ik schrijf 'gaat zijn' want het bestaat op moment van schrijven nog niet. Lang verhaal kort: Web1.0 (1991-2004) bestond voornamelijk uit statische pagina's en was read-only. Bij de opkomst van het huidige Web2.0 (vanaf 2004-heden) zijn users meer interacties gaan maken met het web waardoor grote partijen zoals Google, Facebook en Amazon meer data hebben verzameld wat vervolgens wordt verkocht. Als je ooit Facebook hebt gebruikt, dan heb je eigenlijk toestemming gegeven dat je data wordt gebruikt voor marketingdoeleinden. Web3



Metaverse

wilt voorkomen dat al jouw data ligt bij een hele grote centrale organisatie. Grote organisaties kunnen veel met jouw data doen zoals dit bijvoorbeeld censureren waar zij dit nodig achten.

In het ideale beeld van Web3, behoort data van het web niet bij een grote centrale organisatie en ben jij eigenaar van jouw data of content. Iedereen heeft toegang tot alle informatie van de blockchain en een blockchain identity, die geheel legitiem is want deze identity aanpassen gaat niet zomaar. Ook kan je legitiem digitaal eigenaar zijn van assets op de blockchain, zoals crypto, NFT's en tokens. Plus iedere user heeft een wallet waarmee je assets koopt en in bewaart.



HistoryWeb

Een munt heeft altijd twee kanten en natuurlijk zijn er ook nadelen aan Web3. Het kan onhandig zijn dat data onveranderlijk is en bepaalde dingen niet gecensureerd worden omdat er geen centrale organisatie is die een oogje in het zeil houdt, zoals negatieve discussies op Twitter. Al met al, denk ik persoonlijk, dat Web3 er wel aan zit te komen gezien er veel bedrijven zijn die zich hiermee bezig houden. Tijd om het nu over de Metaverse te hebben!

The Metaverse

Just to be clear: dit staat niet voor een universum waarin Facebook, die zijn naam vorig jaar heeft veranderd in Meta, de regie heeft. De naam Metaverse zou eerder gekozen zijn uit business-en branding motieven en het geloof in de opkomst van de Metaverse: een alternatieve virtuele realiteit waarin mensen gaan werken, spelen en met elkaar connecten. Zoals bijvoorbeeld naar een concert gaan in de game Fortnite. De term Metaverse komt uit een science fiction novel genaamd Snow Crash uit 1992, geschreven door Neal Stephenson.

In de Metaverse is het de bedoeling dat je in de virtuele werelden assets kunt kopen met crypto, je blockchain identiteit gebruikt en alles omtrent data is gedecentraliseerd. Web3 meets the Metaverse! Supercool, toch?

Het is cool als je als frontend developer toffe en mooie dingen bouwt maar het voelt nóg beter als met jouw design ook daadwerkelijk impact maakt.

Dan gaan we nu in waarom jij als frontend developer je zeker wilt bezig houden met blockchain development!

5 redenen waarom je blockchain development wilt leren als frontend developer.

1. Pionieren aan de voorgrond: blockchain, Web3 en de Metaverse zijn in opkomst

Met de komst van Web3, gaat een nieuw tijdperk beginnen voor de manier waarop met onze data wordt omgegaan. Industrieën zoals de gezondheidszorg, banken (ja, zelfs die!) en duurzaamheidsinitiatieven gebruiken blockchain om de betrouwbaarheid, veiligheid en transparantie van hun data te garanderen. Blockchain kan de missie van bedrijven versterken doordat het hen ook tijd, energie en geld kan besparen. Bijvoorbeeld omdat data sneller te vinden is, wijzigingen realtime plaatsvinden en minder resources nodig zijn om bepaalde checks te doen. Als blockchain software developer zou ik dan willen nadenken over hoe het design van apps eruit moet zien zodat blockchain de missie van een bedrijf kan versterken.

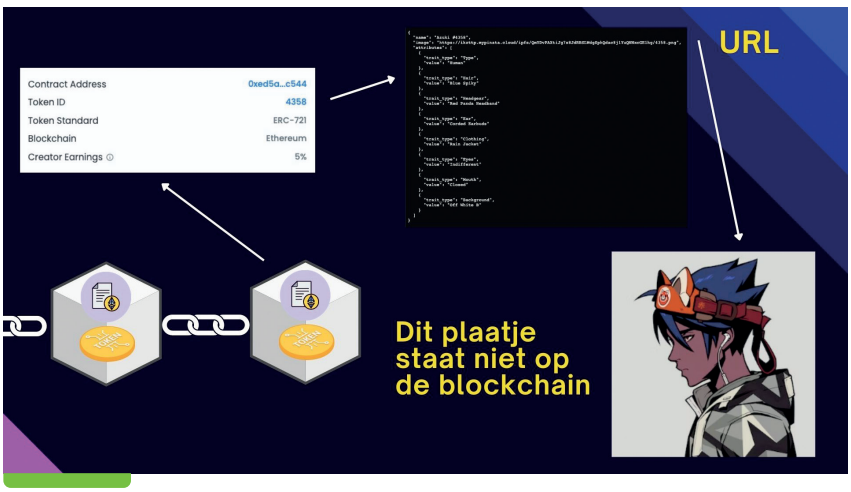
2. De explosie aan creativiteit in de blockchain space

Als frontender ben je nou eenmaal best wel creatief, nietwaar? Als je op Google zoekt naar NFT's of blockchain games, dan moet je bijna wel geïnspireerd worden om zelf creatief aan de slag te gaan. Wat ik het meest fascinerend vind, is dat het bij NFT's en blockchain games het niet alleen om cool design gaat maar ook om het belang van owners en players. Laten we NFT's onder de loep nemen.

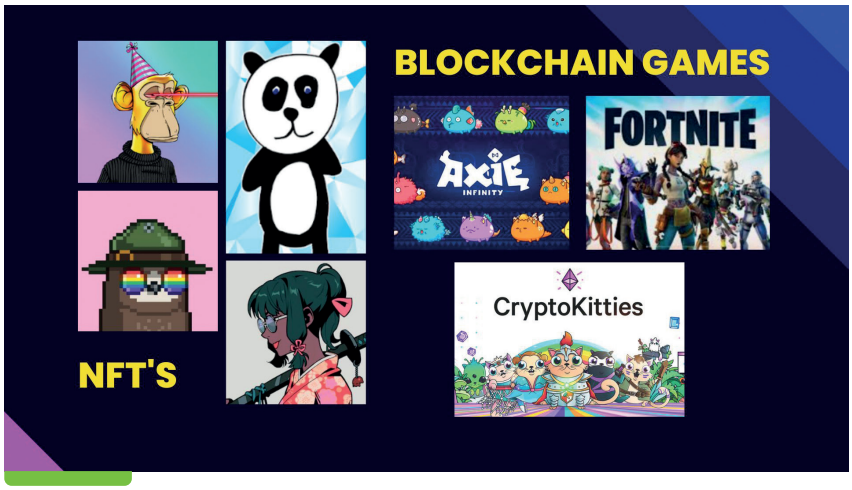
NFT staat voor Non Fungible Token en dat betekent dat deze token iets unieks representeert en niet zomaar kan worden geruild met een andere NFT. Bitcoins bijvoorbeeld zijn juist 'fungible' want een bitcoin is gelijk aan een andere bitcoin. Het gaat bij NFT's voornamelijk erom dat je kan zien wie eigenaar is van unieke assets zoals o.a. digital art, filmpjes of stukken virtueel onroerend goed. Goed om te weten is dat de asset die de token representeert, niet op de blockchain staat. De token heeft alleen een url dat verwijst naar dat stukje kunst. Zo wordt de blockchain niet overvol met kunst maar mensen zouden jouw NFT ook makkelijk kunnen kopiëren omdat de blockchain transparant is.

NFT's...is het een hype of...? Behalve dat er veel geld wordt verdiend aan het verkopen van kunstige JPEGs, gaat het om de onderliggende smart contracts technologie. Dit zijn programma's die gelijk tot executie overgaan op basis van wat er in een smart contract is bepaald. Daar staat bijvoorbeeld in dat Pino de nieuwe eigenaar is van een NFT omdat zij dit voor 0.8 ETH (Ethereum) vandaag van Oscar heeft gekocht. Ook kan in het contract staan dat Ernie de originele kunstenaar is van de NFT en daarom bij iedere verkoop daarvan, een percentage ontvangt van de verkoopprijs. Dit krijgt hij direct wanneer Oscar ETH krijgt van Pino en kan allemaal gebeuren zonder tussenkomst van een derde partij!

Als creatieve frontender, wil je mooie dingen maken en hoe cool is het als je een NFT-marketplace maakt of een NFT-gallery voor digi-



NFTStructuur



NFTsGames

tale artiesten zodat hun werk veilig en goed wordt verkocht?
Plus in blockchain-games draait het vaak om play-to-earn concepten waarbij je door te spelen crypto, in-game items of NFT's kan verdienen. Denk aan de hoeveelheid creativiteit, design en innovatie die gepaard gaat met het bouwen van de game, hun dashboards en virtuele werelden. Zo cool!

3. Veel carrièremogelijkheden en kans op een lucratief salaris

Op basis van onderzoek[2], was het wereldwijde marktaandeel van blockchain technologie in 2021 geschat op 4.8B USD en wordt in 2030 verwacht dat dit rond 69B USD gaat zijn. Dat betekent dat er veel carrièremogelijkheden zijn als blockchain developer in deze sector. Het gemiddelde jaarsalaris van de gemiddelde blockchain developer is 143.000 USD waarbij entry-level start op 121.000 USD en seniors op 175.000 USD.

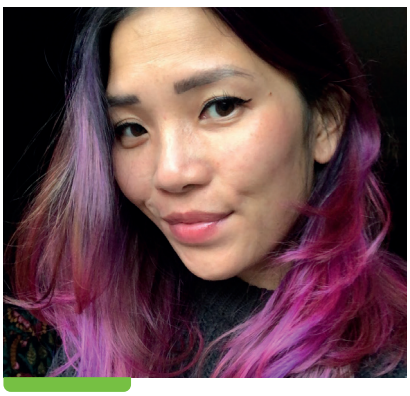
Momenteel zitten we nog in het vroege stadium aangezien Web3 er nog niet is. De vraag naar blockchain developers is momenteel veel[3] groter dan het aanbod. Daarom is het nu de beste tijd om te worden aangenomen als blockchain software developer. Ingeval je hieraan zit te denken, is het aan te raden om eerste te focussen op het leren van Solidity waarmee je smart contracts kan maken. Houd er dan wel rekening mee dat je dan voornamelijk met de Ethereum Virtual Machine (EVM) werkt en blockchains die hiermee kunnen werken. Wat niet erg is want de EVM wordt veel gebruikt!

4. Veel support en communities

Er is op moment van schrijven ontzettend veel support te vinden in communities die zich bezig houden met NFT projecten, Web3-initiatieven en blockchain games. Bijvoorbeeld op LinkedIn zijn groepen te vinden voor blockchain developers en mensen die graag connecten over de Metaverse. Ook worden er veel blockchain Hackathons georganiseerd waarbij mensen bij elkaar komen om de meest innovatieve blockchain app te maken. De community app Discord is bij uitstek een hele goede plek om communities van mensen die in de blockchain space werken.

5. Het maken van impact, wie wil dat nou niet?

Het is cool als je als frontend developer toffe en mooie dingen bouwt maar het voelt nóg beter als met jouw design ook daadwerkelijk impact maakt. Dit kan liggen in het mooi visualiseren van data of dat users beter, sneller en handiger met de app kunnen werken wegens intuïtief design. Of dat je de branding van een artiest zodanig weet vorm te geven op een website dat hij of zij goed NFT's verkoopt.
Wil jij na het lezen van dit artikel je meer verdiepen in blockchain development en frontend? Laat het me dan weten als dit artikel je geïnspireerd heeft of als je vragen hebt over blockchain, Web3 en de Metaverse! ●



Jessy The

Footnote:
0. bitcoin.org/bitcoin.pdf
1. https://decrypt.co/101340/bitcoin-quantum-computing
2. https://www.globenewswire.com/news-release/2022/08/25/2504745/0/en/Global-Blockchain-Technology-Market-Size-Worth-69-Billion-by-2030-at-a-68-CAGR-Check-Blockchain-Industry-Share-Growth-Trends-Value-Analysis-Custom-Market-Insights.html
3. https://101blockchains.com/demand-for-blockchain-skills/

FUTURETECH
TERUGBLIK

Eindelijk was het weer zover! Ons geliefde Future Tech conferentie kon weer fysiek plaats vinden! Na twee jaar het event online te organiseren, konden we weer verzamelen in de Jaarbeurs van Utrecht op 21 juni voor een volgepakt programma met sprekers uit binnen- en buitenland. Een supergezellig sponsorplein waar je t-shirts, sokken, lego wolken, lego bouwpakketten kon ophalen of verdienen, er was van alles mogelijk. Bij de SDN-cast stand kon je mokken en vliegenmeppers winnen door te draaien aan het rad en jou menig te geven voor de camera! Dat hebben veel bezoekers ook gedaan, die compilatie kun je terugvinden op: https://www.youtube.com/watch?v=DvTbtFjdLPw



Het belangrijkste waren natuurlijk de sprekers! We hadden een goed gevuld programma met vele mooie sessies en vele goede sprekers. Sprekers zoals James Ashley, Johny Hooyberghs, Sjoukje Zaal, Henk Boelman, François Bouteruche, Geert van der Cruijsen, Mike Martin, Gerald Versluis, Dennis Doomen, Dennis van der Stelt, Amy Vanderburg, en nog veel meer bekende en minder bekende sterren.

En natuurlijk Scott Hunter; Mister .NET Core himself! Maar het spook van de afgelopen twee jaar speelde ons wel parten. Scott was, op het allerlaatste moment, vlak voordat hij zou afreizen naar Nederland getroffen door Covid en de verschijnselen. We zaten als organisatie met de handen in het haar, hoe gaan we dit opvangen: vervangen of laten vervallen? We hebben gekozen om Scott via een teleconferentie alsnog zijn sessie te laten doen. We wisten dat dit niet de kwaliteit had zoals wij (Future Tech) voor ogen hadden en waar we voor staan. Maar we wisten dat veel bezoekers speciaal voor Mister .NET Scott zouden komen, dus vervangen was een grotere teleurstelling. Scott heeft ondanks de koorts en de afstand zijn uiterste best gedaan om er een mooie show van te maken.



Bij de SDN-Cast stand hebben we verschillende sprekers geïnterviewd en gesproken. Een compilatie daarvan kun je vinden op https://www.youtube.com/watch?v=bzmFzag-byYg. Bas Lijten, Morten Ege Jensen, Robert Jan van Holland, Mike Martin, Niels Tanis, Amber Vanderburg, Eduard Keiholz, Henry Been, Leo Visser, Marloes Bijkerk, Jan de Vries, Menno Jongerius en Annejan Barelds kwamen langs en gingen het gesprek aan met Marcel en Roy. Bijzondere en grappige gesprekken over IT en development in het bijzonder, een leuk uurtje youtube.

We zijn alweer bezig met de editie voor 2023 en we gaan daar een nog mooier en grootser evenement van maken. Hou de datum alvast geblokt in je agenda, lees ons magazine en check onze socialmedia kanalen regelmatig voor meer nieuws en updates. Future Tech 2023 wordt beter, mooier en nog gezelliger!

En kijk vanaf oktober weer twee wekelijks naar de SDN Cast op https://www.youtube.com/SDNCast.


Invent
with
purpose.
ABN·AMRO
WELCOME

Slim automatiseren vanuit één centrale plek in de organisatie

Auteur: Jelle de Graaf

Ken je dat? Je hebt een organisatie van honderden medewerkers van wie velen taken automatiseren. Zij bevinden zich op verschillende afdelingen en hebben weinig tot geen contact met elkaar. Toch zijn er taken die het gros van de teams uitvoert. Zoals de MFA van een eindgebruiker resetten of het aanmaken van een nieuw gebruikersaccount bij een klant. De kans bestaat dan dat scripts voor dit soort taken meerdere keren door verschillende mensen geschreven worden. Dat is zonde.

Het idee: een centrale plek voor scripts

Bij OGD wilden we voorkomen dat mensen dubbel werk doen bij het automatiseren van taken. Dus bedachten we dat er een centrale plek moest komen, waar alle collega's suggesties en scripts kunnen vinden en toevoegen. Dit moet gemodereerd worden door een kernteam, dat besluit of een toevoeging daadwerkelijk zinnig is en wordt doorgevoerd.
Deze centrale plek hebben we bij OGD ontwikkeld en noemen we Omnibus. Het is een verzameling scripts, net zoals in de literaire wereld een omnibus een verzameling romans is. Het bijzondere aan deze scripts is dat ze generiek zijn. Ze zijn dus zo geschreven, dat ze geschikt zijn voor gebruik bij iedere klant van OGD. Op die manier richten we nu die ene standaardwijziging die bij elke klant wel voorkomt in één keer goed in en voeren we de wijziging vervolgens bij al die klanten naadloos uit.
Een andere betekenis van omnibus is 'door allen'. Ook met deze betekenis identificeren onze medewerkers zich. Het inner-sourcingkarakter zorgt ervoor dat mensen uit verschillende lagen van de organisatie bijdragen. Van scripts toevoegen tot documentatie schrijven en van een bug aanmelden tot suggesties aanbrengen voor de architectuur.

Kernteam

Het kernteam, geflankeerd door een aantal enthousiaste medewerkers uit andere teams, ontwikkelde Omnibus en leidt het in goede banen. In dit team zitten OGD'ers die gespecialiseerd zijn in het automatiseren van interne processen en taken. Naast het ter beschikking stellen van de techniek, vormt kennisoverdracht ook een groot punt op de agenda. Het automatiseren van veelvoorkomende taken is primair, maar daarmee enthousiasmeren we ook collega's om hetzelfde te doen en betrokken te zijn bij het project.

Gebruikte technieken

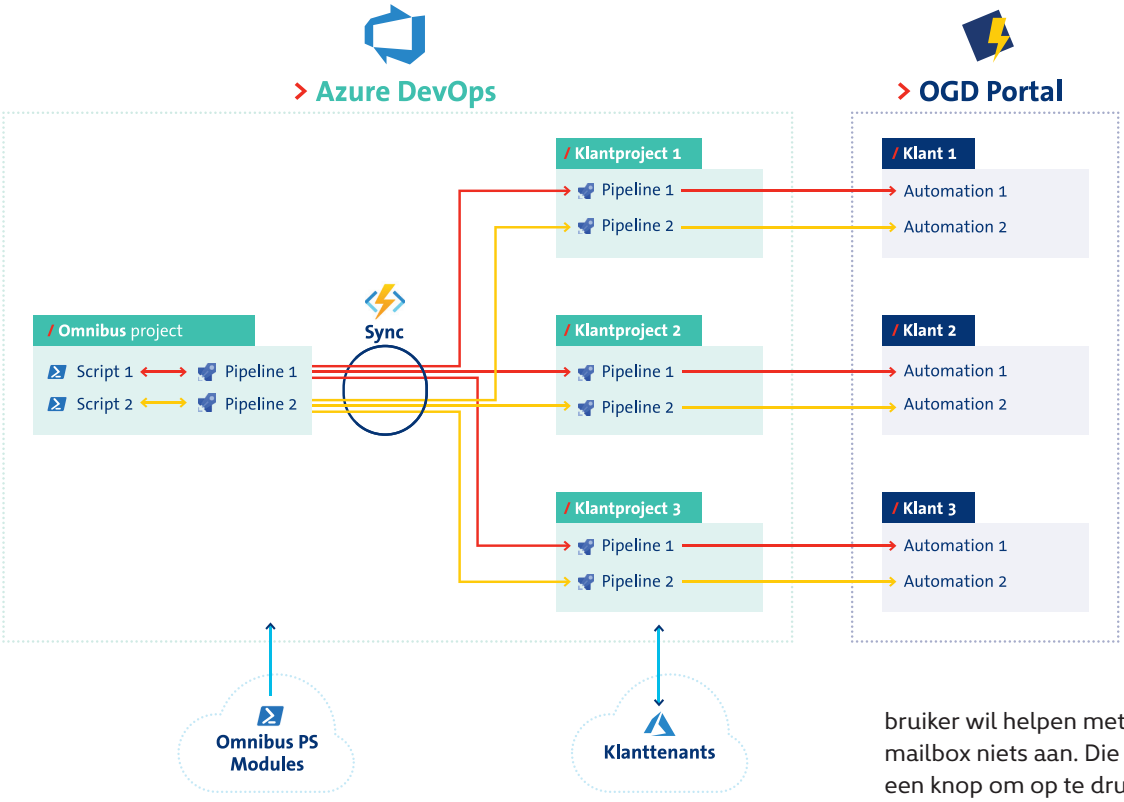
We gebruiken verschillende technieken om alles uit Omnibus te halen. In de back-end staan de diverse scripts (functions) in meerdere PowerShell-modules in onze eigen private PowerShell gallery. Deze functions combineren we tot doelgerichte scripts in repositories op Azure DevOps. Via Azure pipelines worden

BIO

Jelle de Graaf

Jelle de Graaf is een ervaren automation developer bij OGD ict-diensten. Samen met zijn Service Automation team bouwt hij tools en applicaties waarmee hij werk binnen de hele organisatie automatiseert. Hij is gespecialiseerd in Azure Pipelines, Azure Functions en PowerShell. Jelle bouwt graag oplossingen die tijd besparen en snel waarde opleveren. Met zijn team bouwde hij naast Omnibus bijvoorbeeld JIT Admin. Hiermee vraag je een admin account aan dat alleen openstaat wanneer je het actief gebruikt.
Naast zijn standaard werkzaamheden deelt Jelle graag zijn kennis, als spreker en door collega's les te geven. Ook is hij mede-eigenaar van Caravan Escape, dat een caravan heeft omgebouwd tot een mobiele escape room voor evenementen en bruiloften.





de scripts aangeboden als uitvoerbare automations. Deze (nog steeds generieke) automations worden vervolgens weer aangesproken door wrapper pipelines, die in klantspecifieke Azure DevOps projects staan. Deze projects hebben een service connection richting de tenant van de klant, waarmee de authenticatie geregeld wordt. Alle specifieke data die nodig is voor het uitvoeren van de automation, wordt opgegeven bij het starten van de automation (bijvoorbeeld: accountnaam), is al aanwezig in het Azure DevOps project (bijvoorbeeld: service connection), of staat in een variable group (bijvoorbeeld: welke service connection of agent pool er moet worden gebruikt). Tot slot worden deze wrapper pipelines weer via de zelfontwikkelde OGD Portal aan onze servicedeskers, beheerders en specialisten aangeboden.

Kwaliteitscontrole

Omdat de hele organisatie gebruik kan maken van Omnibus, moet er natuurlijk een goede kwaliteitscontrole zijn. In eerste instantie bewaakt het kernteam de toevoegingen. Hoewel iedereen suggesties kan doen, is niet alles bruikbaar. Sommige suggesties zijn bijvoorbeeld te specifiek voor een bepaalde use case of klant om te hergebruiken bij andere klanten. Wanneer een idee wél wordt uitgewerkt, houden we tijdens het ontwikkelen al rekening met alle zaken die nodig zijn om goed te passen in het framework. Zo stellen we eisen aan de code (style guide), de documentatie (volledige comment-based help) en tests (pester tests voor alle functions). De meeste van deze eisen worden met een CI/CD-pipeline gecontroleerd. Nog vóórdat de code live wordt gezet vangen we ze af als er iets niet in orde is. Tot slot voeren we voor het daadwerkelijk mergen van de code nog enkele integration tests uit, om te zien of het geheel waar de wijzigingen onderdeel van uitmaken nog naar behoren werkt.

Front end

Er zit een boel complexe techniek achter de schermen, maar daar heeft een servicedesker die een eindgebruiker wil helpen met een nieuwe mailbox niets aan. Die wil gewoon een knop om op te drukken, die doet wat hij of zij wil. Die knoppen zijn gemaakt in de OGD Portal. In deze Portal krijgt iedereen de klanten te zien waar ze rechten op hebben. Onder deze klanten zien ze vervolgens weer de automations waar ze rechten op hebben.
Rechtenbeheer op dit niveau is belangrijk, omdat het mes aan twee kanten snijdt: je kunt mensen met weinig rechten en ervaring toegang geven tot handelingen die ze anders niet zouden kunnen uitvoeren, maar als je alles voor iedereen beschikbaar maakt, is een foutje of goed bedoelde handeling die niet wenselijk is snel gebeurd.

Conclusie

Met een centrale plek binnen je organisatie voor generieke scripts voorkom je dubbel of zelfs veelvoudig werk. Daarnaast voer je standaardwijzigingen makkelijker, sneller en uniform uit, intern en bij klanten. Het is daarbij wel belangrijk om zo'n verzameling scripts in goede banen te leiden, zowel op het gebied van kwaliteit als op het gebied van rechtenbeheer. Op die manier haalt je organisatie het meeste uit automatiseren ●

Why we suck at estimation

Yesterday it happened again. On my Twitter timeline. Someone angrily posted "developers should finally learn how to estimate better." Another case of work not being done at the end of a sprint. Or worse, another project that didn't make its deadlines.

Auteur: Sander Hoogendoorn

There isn't a single developer out there who hasn't run into similar experiences.
After another deadline not met, another manager argues that we are unprofessional. "I don't understand that you call yourselves professionals, but that you can't even properly estimate the time necessary to implement features X, Y, or Z."
Then they take a short pause for the dramatic effect, followed by the ever popular "How hard can it be?" and the inevitable "When I take my car to the garage, the mechanics know exactly how long it takes to switch my tires. Developers are much higher educated, so why can't you estimate your work?"
And as usual, we feel guilty. Because we missed another deadline, that was based on our estimates. Because yes, we are professionals. So why is it we can't seem to estimate our own work? People in almost every other industry can.
We shrug our shoulders and promise to do better in the next sprint. Or in the next project.
But in all honesty, should we really feel so ashamed that we can't seem to estimate our work? Well, yes and no. As the Scrum people state: "it depends". Let me introduce a model that helps to explain why we sometimes can and sometimes can't. It's called Cynefin.

Let me try to illustrate it really briefly. The Cynefin framework consists of a number of contexts and describes the relationship between problems in particular contexts, and the proper strategies to approach these problems. In a clear context, there is an apparent best practice for the problem at hand. You simply apply the best practice. Done. These are the problems you've dealt with before. Changing your tires in a garage. Or, building a simple web application. Code you've written before. This you can estimate. In a complicated context, there are multiple good practices. Let's say you are selecting a solution for identity access management. There are quite a few good ones. There are people, possibly even in your own company, who've implemented one before. So, you need to select one or two solutions, try them out and implement one. This is still plannable, and you should be able to give a fair estimate. Then it gets harder. When a problem is in a complex or chaotic context, best or even good practices don't exist. At best, they are evolving. You may find people on the internet who've dealt with this before. Or even worse, you can't find anything at all.
When I suggested to the CEO of an insurance company in early 2014 that a microservices architecture would be the best solution for replacing

BIO

Sander Hoogendoorn:

Sander is an independent dad, speaker, writer, and traveler. He has been writing code since 1984 and still codes every day. He is a serial CTO, currently at e-commerce iBOOD. Sander is a code philosopher, agilist, and operated as Capgemini's global agile thought leader for a decade before going freelance in 2015. Following his adagio that small steps are the fastest way forward, Sander helps to empower organizations, teams, and individuals, and to disrupt their ways of working, technology, architectures, and code. He has authored various books and published tons of articles. Sander is a well-known and inspiring keynote speaker at international conferences on diverse topics such as disruption, culture, (beyond) agile, continuous delivery, microteams, monads, software architecture, microservices, and writing beautiful code. Tools do not solve problems, thinking does.





30 million lines of COBOL and Java code, nobody had done microservices before. We could not build on other people's experiences. The only way out was to experiment, and hopefully let practices emerge. If they don't, you need to experiment more.

As you can imagine, problems that live in complex or chaotic contexts are impossible to estimate well. No story points here. At best, you might present a ballpark figure. "We hope we can pull this of in three or four months." Unfortunately, too many managers won't be happy with your ballpark figures. And this is why we suck at estimation. We are dealing with problems in complex or chatoic contexts, but are expected to estimate our experiments as if our problems are in complicated or clear contexts. This is where we need to learn to hold our ground. Explain your managers that you are in the realm of the unknown-unknown - the things you don't know that you don't know. Present them with the Cynefin framework, and explain that you are doing things that nobody you know has done before, and therefore, that you need to experiment. And therefore, that metaphors managers eagerly apply from manufacturing, construction or any other repetitive industry (where work is at worst in a complicated context), do not apply (yet).

But you can also give them some hope. There is some light at the end of the tunnel. After practices emerge, you might be able to push the problems to a complicated context. Where work becomes more predictable. And where you will be able to estimate better.

So, what about the microservices architecture we started to work on in early 2014. Well, it took us almost two years to discover the practices we needed to apply. We started over with parts of the platform multiple times. But eventually, we got good at it. The program finished after almost five years. Successfully. ●



Deel jouw IT kennis **met kinderen!**

Het doel is duidelijk, de kinderen van Nederland meer IT kennis meegeven. Ze krijgen op deze manier een beter beeld hoe mooi onze IT wereld is!

Auteur: Koen Aerts

De stichting Devoxx4Kids is opgericht met dit doel en organiseert al jaren Devoxx4Kids dagen. Het bereik van deze Devoxx4Kids dagen is beperkt (50 kinderen per dag) en tijdens de Corona periode was het in zijn geheel niet mogelijk.
Om de schaalbaarheid te vergroten hebben we een nieuw initiatief opgezet. LITTIL is een online platform waar scholen en IT vrijwilligers (LITTIL docenten) elkaar kunnen vinden! Het initiatief is volledig belangeloos en opgezet vanuit maatschappelijke betrokkenheid. Alle mensen die bijdragen aan dit initiatief (LITTIL docenten, bouwers van het platform, etc.) doen dit op vrijwillige basis. Kinderen een goede en gedegen toekomst meegeven staat hierbij centraal!
Scholen zijn erg enthousiast over dit platform en IT specialisten willen graag hun kennis delen. Dit platform gaat zo'n succes worden dat alle kinderen extra IT kennis gaan krijgen in de toekomst.
Na het lezen van dit artikel weet je alles over LITTIL en kun je jezelf inschrijven op www.littil.org!

Doel

Zoals aangegeven willen we kinderen meer IT kennis meegeven. De IT wereld is de afgelopen decennia enorm gewijzigd. Dit betekent een verscheidenheid aan software en ook een diversiteit aan rollen binnen onze sector.
Naast software op de laptop en op de telefoon, werken er oneindig veel apparaten tegenwoordig met software. Software development voor websites is compleet verschillend van software development op embedded systemen (bijvoorbeeld auto's).
Het is belangrijk om de kinderen een goed beeld te geven van onze IT wereld zodat ze een weloverwogen beroepskeuze kunnen maken. Kinderen hoeven zeker niet allemaal voor het IT vak te kiezen, als ze maar weten waar ze ja of nee tegen zeggen.

LITTIL

De Devoxx4Kids stichting is al diverse jaren actief om kinderen meer IT kennis mee te geven. Speciale Devoxx4Kids dagen worden georganiseerd waar ongeveer 50 kinderen per keer aan deel kunnen nemen. Jaarlijks werden er voor de Corona pandemie

Alle mensen die bijdragen aan dit initiatief, doen dit op vrijwillige basis. Kinderen een goede en gedegen toekomst meegeven staat hierbij centraal

ongeveer 10 dagen per jaar georganiseerd waardoor er 500 kinderen op jaarbasis bereikt werden. De schaalbaarheid van deze opzet kent zijn beperkingen.

Tijdens Corona is er een nieuw concept ontstaan, een platform wat scholen (zowel basis als middelbare scholen) en IT vrijwilligers samenbrengt. Het platform LITTIL biedt naast deze functionaliteit ook nog een derde as, er zijn LITTIL lesmodules beschikbaar die je kunt gebruiken. Door deze LITTIL modules wordt het laagdrempelig voor de IT vrijwilligers om zich in te schrijven en als LITTIL docent te gaan werken. Op deze manier kunnen jullie al vanaf één vrije dag helpen als LITTIL docent. Door het platform wordt alles beschikbaar gesteld om op een verantwoorde manier maatschappelijk betrokken bezig te zijn!

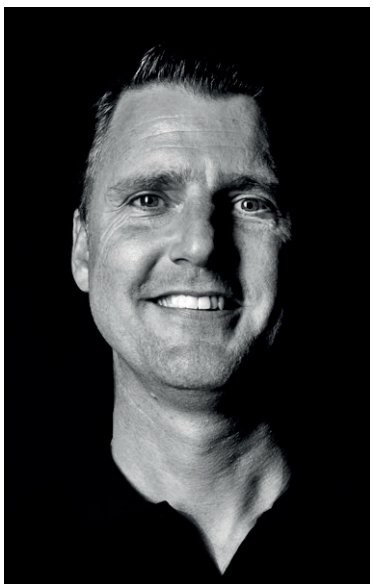
Inclusiviteit

De verhouding man/vrouw binnen de IT sector is in behoorlijk veel landen gelijk verdeeld. Denk hierbij aan Oost Europa of India. Er zit iets diepgeworteld in onze Nederlandse maatschappij waardoor de verhouding bij ons behoorlijk anders is, deze gaat meer naar 90% man en 10% vrouw. Dit terwijl er geen argumenten zijn waarom mannen beter zouden kunnen programmeren dan vrouwen. Door kinderen in zijn algemeenheid enthousiast te maken voor IT willen we ook meisjes meenemen om tegelijkertijd deze scheefgroei aan te kunnen pakken. Sterker nog, LITTIL staat voor inclusiviteit op alle gebieden. Speciale focus willen we ook geven aan kinderen die in hun eigen milieu minder met IT in aanraking komen. Door de opzet van LITTIL verwachten we dat we ook beter met deze groep in contact kunnen komen. ●

BIO

Koen Aerts

Koen Aerts is CTO bij Team Rockstars IT. Na diverse jaren als software ontwikkelaar en architect gewerkt te hebben heeft hij de stap naar management functies gemaakt. Koen is erg innovatief en creatief en is actief in veel software communities. Daarnaast is hij voorzitter van de Devoxx4Kids stichting.



DEVOXX™ 4KIDS Netherlands

Ook de allerjongste liefhebbers van IT **hebben de toekomst!**

Auteur: Sanneke van der Meer

Wij geloven dat je door samen te werken, samen beter wordt. Daarom delen we onze ervaring zo breed mogelijk. Dit doen we bij vakgenoten, maar we willen ook de nieuwe generatie enthousiast maken voor ons mooie vak. Wij werken samen met CoderDojo Nederland om zo veel mogelijk jonge mensen te bereiken.

Wie is CoderDojo

Bij CoderDojo leren kinderen tussen de 7 en 17 jaar programmeren, apps of websites maken en meer. De bijeenkomsten ('dojo's') zijn altijd gratis en worden verzorgd door enthousiaste mentoren. CoderDojo draait volledig op vrijwilligers. CoderDojo Nederland ondersteunt de landelijke community via het delen van kennis, organiseren van meet-ups en koppelen van vraag en aanbod. Met deze activiteiten zorgt CoderDojo Nederland ervoor dat de Dojo's in het land een zo groot mogelijke impact hebben op de maatschappij.

Dit zegt Sanneke van der Meer over CoderDojo

Eind 2016 startte ik met het organiseren van CoderDojo in de bibliotheek. In elke Dojo (en we organiseren er sindsdien bijna elke week één op zeven verschillende locaties!) zien we kinderen leren en groeien. Het maakt niet uit of het hun eerste of hun 100e is. Ze leren robots programmeren zoals Dash en Leaphy, we hebben Micro:bits, Arduino's en Raspberry Pi's en ze leren websites, games en apps bouwen.
De kinderen ontmoeten gelijkgestemde kinderen en krijgen hulp van onze fantastische mentoren; iedereen werkt met elkaar samen! Als ik de trotse gezichten van de kinderen zie aan het einde van de sessies, dan geniet ik daar elke keer weer van.

Wat is jullie grootste uitdaging?

Onze grootste uitdaging op dit moment is om meer meiden naar de Dojo's te trekken. Jongens zijn nog steeds in de meerderheid. Gelukkig is 50% van onze mentoren vrouw, dus we geven hiermee al wel het goede voorbeeld! In de afgelopen maanden is het aantal meiden toegenomen. We geven ook programmeerlessen op scholen en dat helpt om meiden te interesseren om een Dojo te bezoeken. Het blijft lastig als je ouders tegen hun dochters hoort zeggen: "dat is niets voor jou!". We doen er als bibliotheek van alles aan om ouders van gedachten te laten veranderen!
Wil je meedoen en jonge mensen enthousiast maken? Kijk eens op https://coderdojo.nl/ wat het betekent om mentor te worden bij een Dojo bij jou in de buurt.

Leuke projecten om in de gaten te houden

Elk jaar organiseert Esero twee mooie projecten waar je vanuit een Dojo of met je klas aan mee kunt doen. In beide gevallen programmeer je met de jongeren een module die gaat draaien op de astro-pi in het ISS ruimtestation.
Voor kinderen op de basisschool van groep 6 t/m 8 is er het **Mission Zero project**. Bij Mission Zero schrijven leerlingen een programma waarbij de Astro Pi op het LED scherm een boodschap laat zien. In die boodschap moet in ieder geval de luchtvochtigheid in het ISS zijn verwerkt. https://esero.nl/projecten/astro-pi/sub-pagina/mission-zero
Voor jongeren tot 19 jaar is er het **Mission Space Lab** project. In Mission Space Lab zetten leerlingen een experiment op dat de Astro Pi in het ISS kan uitvoeren. Ze kunnen kiezen uit twee thema's: Leven in de ruimte, en Leven op aarde. Ze kunnen metingen verrichten aan temperatuur, luchtdruk, luchtvochtigheid en een bewegingssensor, of een foto van het aardoppervlak in infrarood maken. https://esero.nl/projecten/astro-pi/sub-pagina/meedoen-met-mission-space-lab
Dus hele mooie initiatieven om jonge mensen enthousiast te maken. We hopen jou ook een keer te ontmoeten op een Dojo in de buurt..





> Denk jij dat je het beter weet?

/ Mooi, dan zoeken we jou!

/ Werk aan complexe ict-projecten
én deel jouw kennis met 1300 collega's.

> werkenbij.ogd.nl
Je baan is meer dan werk.

Modulaire opzet in Blazor Webassembly

Auteur: Menno Jongerius

Nu Blazor steeds prominenter in het Microsoft landschap terechtkomt, zie ik de eerste teams productie applicaties opzetten in Blazor. Eén van de uitdagingen die je daarbij hebt, is hoe je de nieuwe applicatie structureert, zodat hij kan groeien zonder dat het een chaotische brij wordt. In de back-end systemen hebben we tal van mogelijkheden om applicaties op te knippen. In de front-end zijn die opties lastiger. Ik hoop in dit artikel een paar handvatten te geven om je Blazor client modulair op te zetten.

Modulair versus "Micro" service

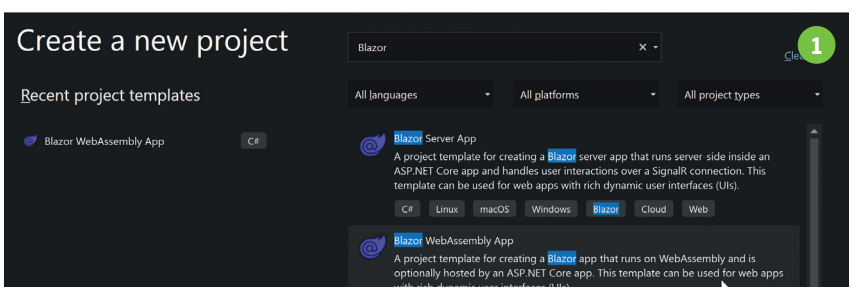
De aanpak die ik in dit document beschrijf leidt tot modules in een single page Blazor client. Dit levert meerwaarde in structuur en maakt het ook gemakkelijker om eventueel met meerdere teams aparte modules door te ontwikkelen. Alleen de uiteindelijke verwerking tot één single page client, maakt de applicatie per definitie monolithisch van aard. Dat gaat wringen als de client niet langer een mooi samenhangend geheel aan functionaliteit biedt. Als dat het geval is, overweeg dan om meerdere gespecialiseerde clients op te zetten, of duik eens in de mogelijkheden van piral (https://piral.io/), een framework om micro services in de front-end op te zetten. Voor dit document focus ik enkel op een beter gestructureerde "monoliet" in Blazor.

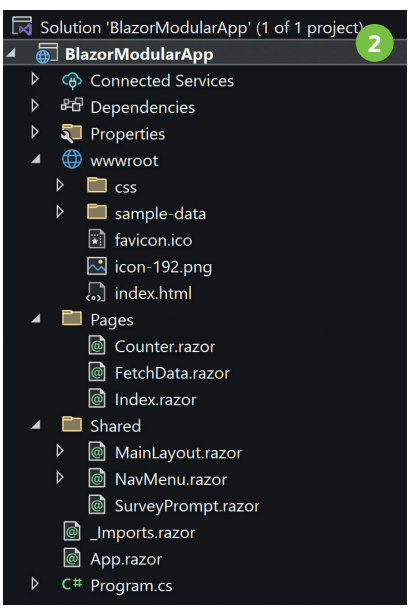
Basis opzet Blazor client template

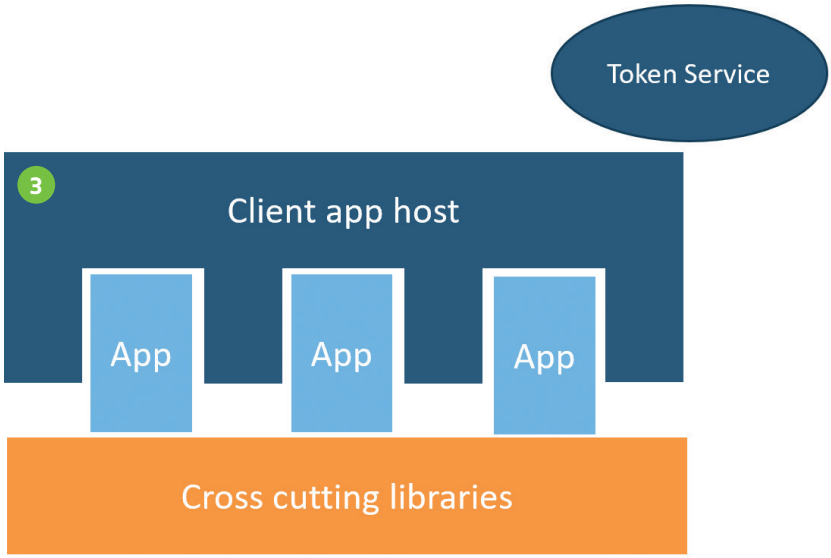
Om een modulaire Blazor client op te zetten, starten we met een standaard Blazor client template: **Figuur 1**.
Elke nieuwe client heeft dezelfde opbouw. De wwwroot bevat statische bestanden van het nieuwe client project. Je moet dan denken aan afbeeldingen, javascript, css en eventuele statische html pagina's. Standaard staat hier ook de index.html als startpagina van de applicatie.
Naast de wwwroot staan de mappen en bestanden die gecompileerd worden tot je applicatie. **Figuur 2**.
De program.cs is, zoals bij de veel dotnet templates, het startpunt van de applicatie. De app.razor bevat router informatie van de applicatie. Deze informatie zorgt ervoor dat de applicatie naar de juiste pagina navigeert op basis van de url in de browser.
De shared map bevat gedeelte (partial) views en layouts en de pages bevatten de razor pages van de applicatie. In de standaard opzet zitten code en de razor pages bij elkaar in één bestand. Eigenlijk elk team dat ik ken, splitst dat op in aparte bestanden voor de code (viewmodels) en de razor syntax (views). Daarmee volgt Blazor het MVVM model zoals je mogelijk van andere templates zoals WPF kent.
Je kunt je voorstellen dat deze standaard opzet van de applicatie al snel ongecontroleerd groeit, naarmate het aantal pagina's toeneemt, omdat functionaliteit verdeelt kan zijn over razor pagina's, mogelijke statische bestanden, gedeelde componenten enzovoorts.

Een modulaire opzet opbouwen

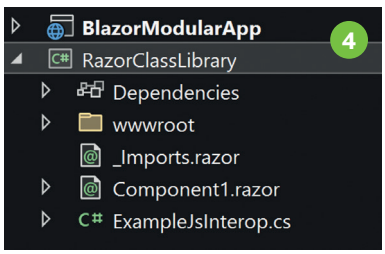
De modulaire opzet die we in dit project kiezen zorgt ervoor dat de Blazor client applicatie als een schil draait, terwijl de functionele componenten ieder hun eigen module krijgen: **Figuur 3**.
In deze opzet zou je zover kunnen gaan dat aparte teams aan een module werken, maar ook als je maar één







team hebt helpt het om je applicatie gestructureerd te houden. Er is in Visual Studio een standaard template om dit soort modules te bouwen voor Blazor: de "Razor class library". Door een dergelijk project aan je solution toe te voegen komt je structuur er als volgt uit te zien: **Figuur 4**.
Je ziet dat de nieuwe class library zijn eigen wwwroot heeft voor statische files, en je ziet dat er al een standaard razor component is toegevoegd. De wwwroot van de module en van het client project worden tijdens het bouwen of publiceren samengevoegd.



Uitdaging 1: Loosly coupled modules

Als je met losse modules werkt, wil je voorkomen dat je het hoofdproject moet aanpassen bij elke aanpassing in de module. Om dat te doen, moet de module zoveel als mogelijk losgekoppeld zijn van de clientapplicatie. De standaard program.cs van de clientapplicatie ziet er als volgt uit: **Figuur 5**.
Je ziet hier dat bij builder.Services de registratie van objecten plaatsvindt. Meestal zal dit bij grotere projecten al lang verplaatst zijn naar een aparte plek in de applicatie, maar wat voor onze module belangrijk is, is dat je niet in deze program.cs wijzigingen wilt doorvoeren voor elke wijziging in de module. **Figuur 6**.
Om dat voor elkaar te krijgen voegen we in de module, AppOne in dit voorbeeld, een class toe om de IOC registraties te doen. In dit voorbeeld is het een extension method, maar dat hoeft natuurlijk niet. **Figuur 7**.
In het hoofdproject is het nu voldoende om de registratie van AppOne aan te roepen. Vanaf dit moment kan AppOne groeien zonder dat er in de registratie in het hoofdproject iets hoeft te wijzigen. Bijkomend voordeel is dat objecten die alleen voor AppOne bedoeld zijn, als internal gedeclareerd kunnen worden.

Uitdaging 2: Routing

Binnen Blazor declareer je de routing meestal als decorator op de razor view: **Figuur 8**.
In de app.razor initialiseer je vervolgens de routing: **Figuur 9**.
Standaard zie je dat de routing alleen de assembly van het hoofdproject inleest. Het is vrij eenvoudig om de assemblies van de verschillende modules hieraan toe te voegen, maar het

5
```
var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");

builder.Services.AddScoped(sp => new HttpClient { BaseAddress = new Uri(builder.HostEnvironment.BaseAddress) });

await builder.Build().RunAsync();
```

6
```
namespace AppOne
{
    0 references | Menno Jongerius, 27 days ago | 1 author, 2 changes
    public static class Setup
    {
        1 reference | 0 changes | 0 authors, 0 changes
        public static IServiceCollection AddAppOne(this IServiceCollection services)
        {
            services.AddScoped<ExampleJsInterop>();

            return services;
        }
    }
}
```

```
builder.Services.AddAppOne();
```
7

```
@page "/AppOne"
```
8

```
<Router AppAssembly="@typeof(App).Assembly">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
        <FocusOnNavigate RouteData="@routeData" Selector="h1" />
    </Found>
    <NotFound>
        <PageTitle>Not found</PageTitle>
        <LayoutView Layout="@typeof(MainLayout)">
            <p role="alert">Sorry, there's nothing at this address.</p>
        </LayoutView>
    </NotFound>
</Router>
```
9

betekend wel een extra plek in de code waarop in het hoofdproject een wijziging nodig is als er een nieuwe module wordt gemaakt. Het aantal plekken waarop het hoofdproject wijzigt bij een wijziging in modules wil je tot een minimum beperken. Voor dit voorbeeld kies ik een eenvoudige opzet die in ieder geval het principe duidelijk maakt van zo min mogelijk afhankelijkheden.

Mijn praktisch doel is dat de enige plek waar een module aangemeld moet worden, de program.cs van het hoofdproject is. Dat betekend dat ik een interface nodig heb die zowel in het hoofdproject als in de module bekend is. Met andere woorden, we hebben een eerste cross-cutting functionaliteit nodig die in een gedeelde library komt: **Figuur 10**.

```
namespace SharedLibs.Interfaces
{
    8 references | Menno Jongerius, 83 days ago | 1 author, 1 change
    public interface IAssemblyRegistration
    {
        2 references | 0 changes | 0 authors, 0 changes
        void RegisterAssembly(Assembly assembly);

        2 references | Menno Jongerius, 83 days ago | 1 author, 1 change
        Task<IEnumerable<Assembly>> GetRegisteredAssemblies();
    }
}
```
10

```
namespace SharedLibs.Services
{
    1 reference | Menno Jongerius, 83 days ago | 1 author, 1 change
    public class AssemblyRegistrationService : IAssemblyRegistration
    {
        private readonly List<Assembly> _assemblyList = new List<Assembly>();

        2 references | Menno Jongerius, 83 days ago | 1 author, 1 change
        public Task<IEnumerable<Assembly>> GetRegisteredAssemblies()
        {
            return Task.Run(() =>
            {
                return _assemblyList.AsEnumerable();
            });
        }

        3 references | 0 changes | 0 authors, 0 changes
        public void RegisterAssembly(Assembly assembly)
        {
            _assemblyList.Add(assembly);
        }
    }
}
```
11

```
var assemblyRegistrationService = new AssemblyRegistrationService();
var builder = WebAssemblyHostBuilder.CreateDefault(args);
builder.RootComponents.Add<App>("#app");
builder.RootComponents.Add<HeadOutlet>("head::after");

builder.Services.AddSingleton<IAssemblyRegistration>(sp => assemblyRegistrationService);

builder.Services.AddAppOne(assemblyRegistrationService);
```
12

De assembly registratie implementatie kunnen we heel eenvoudig houden voor het doel wat we voor ogen hebben: **Figuur 11**.
De program.cs van het hoofdproject passen we eenmalig aan om de nieuwe assembly registratie toe te voegen: **Figuur 12**.
Je ziet hier een wat afwijkende vorm van registratie van de IAssemblyInterface. In dit geval wordt er bij het opstarten van de applicatie een instantie van de AssemblyRegistrationService geïnstantieerd. In de service provider wordt geregistreerd dat deze instantie van de registration service altijd teruggegeven moet worden indien deze opgevraagd wordt. De reden om dit op deze manier te doen, is dat de AssemblyRegistrationService al gebruikt wordt tijdens het initialiseren van de modules bij AddAppOne. We hebben dus al vroeg in het proces een concrete implementatie nodig.

BIO

Menno Jongerius

CTO Bergler, helpt software teams in .NET #SOLID. Ik werk graag op het snijvlak van mensen en techniek. Ik geloof dat geweldige software tot stand komt wanneer gedreven professionals samen werken. Ik draag hier mijn steentje aan bij door teams te helpen en motiveren, door kennis te delen en mensen met elkaar in contact te brengen.



```
public class AssemblyRegistrationService : IAssemblyRegistration
{
    private readonly List<Assembly> _assemblyList = new List<Assembly>();
    private readonly List<IModuleRegistrationInfo> _modulesRegistrationInfo = new List<IModuleRegistrationInfo>();


    1 reference | Menno Jongerius, 89 days ago | 1 author, 1 change
    public Task<IEnumerable<Assembly>> GetRegisteredAssemblies()
    {
        return Task.Run(() =>
        {
            return _assemblyList.AsEnumerable();
        });
    }

    1 reference | Menno Jongerius, 89 days ago | 1 author, 1 change
    public IEnumerable<IModuleRegistrationInfo> GetRegisteredModules()
    {
        return _modulesRegistrationInfo.AsEnumerable();
    }

    3 references | Menno Jongerius, 89 days ago | 1 author, 1 change
    public void RegisterAssembly(Assembly assembly, string moduleName, string moduleLink, string moduleIcon)
    {
        _assemblyList.Add(assembly);
        _modulesRegistrationInfo.Add(new ModuleRegistration
        {
            Name = moduleName,
            Link = moduleLink,
            Icon = moduleIcon
        });
    }
}
```

13

je de client applicatie onafhankelijk maakt van de module. In de setup van AppOne registreren we nu naast de assembly ook wat info voor de navigatie: **Figuur 14**.

Als laatste passen we de hoofdnavigatie aan om de module linkjes te renderen: **Figuur 15**.

Je ziet hier dat de pagina automatisch voor elke geregistreerde module een hoofdnavigatie link aanmaakt. Dit concept kun je volledig uitbreiden voor de use case die je voor ogen hebt.

```
1 reference | Menno Jongerius, 34 days ago | 1 author, 2 changes
public static IServiceCollection AddAppOne(this IServiceCollection services, IAssemblyRegistration assemblyRegistration)
{
    assemblyRegistration.RegisterAssembly(typeof(Setup).Assembly, moduleName: "App One", moduleLink: "AppOne",ModuleIcon: "box");
```

14

```
<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
    <nav class="flex-column">
        <div class="nav-item px-3">
            <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
                <span class="oi oi-home" aria-hidden="true"></span> Home
            </NavLink>
        </div>
        <div class="nav-item px-3">
            <NavLink class="nav-link" href="counter">
                <span class="oi oi-plus" aria-hidden="true"></span> Counter
            </NavLink>
        </div>
        <div class="nav-item px-3">
            <NavLink class="nav-link" href="fetchdata">
                <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
            </NavLink>
        </div>
        @if (modules == null)
        {
            <p><em>Loading...</em></p>
        }
        else
        {
            @foreach (var module in modules)
            {
                <div class="nav-item px-3">
                    <NavLink class="nav-link" href="@module.Link">
                        <span class="oi oi-@module.Icon" aria-hidden="true"></span> @module.Name
                    </NavLink>
                </div>
            }
        }
    </nav>
```

15

In het model vul je de lijst met modules vanuit de AssemblyRegistration en klaar is kees: **Figuur 16**.

```
private IEnumerable<IModuleRegistrationInfo>? modules;

protected override void OnInitialized()
{
    modules = AssemblyRegistration.GetRegisteredModules();
}
```

16

Eindresultaat

Na deze stappen hebben we een opzet gemaakt van onze Blazor client waarbij de client applicatie zelf als host functioneert en alle functionaliteit in modules is gezet. Elke module kan zoveel pagina's bevatten als nodig is, en doordat elke module zijn eigen wwwroot heeft kunnen ook statische bestanden bij de module worden opgeslagen. Gedeelde componenten worden in de client applicatie zelf, of in de shared library opgenomen. **Figuur 17**.

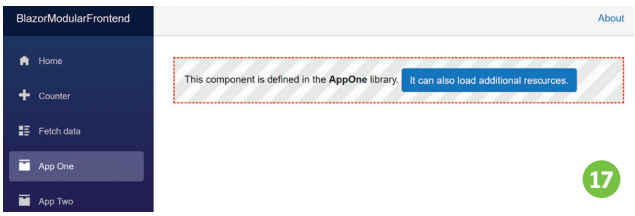
Uitdaging 3: Navigatie

In dit voorbeeld gebruik ik de default implementatie van de navigatie die je meekrijgt in de template. In de praktijk ziet je navigatiemenu er waarschijnlijk heel anders uit. Het principe blijft echter gelijk: we willen niet dat we in de navigatie aanpassingen moeten doen bij de registratie van een module. Dit betekend dat we in de modules een navigatie structuur moeten definiëren die we kunnen meegeven aan de client applicatie, of die we in de module zelf in de html zetten. In dit voorbeeld breid ik de AssemblyRegistrationService uit met aanvullende functionaliteit om een navigatie item in het hoofdmenu te zetten: **Figuur 13**.

Je kunt de strings in deze implementatie lelijk vinden, maar het gaat me hier enkel om het principe dat

Ja maar ...

Natuurlijk kom je in de praktijk nog veel meer uitdagingen tegen, maar dan zou dit artikel een boek worden. Ik hoop vooral dat dit je een richting geeft die je gaat helpen om je Blazor client beter te structureren. ●



17

Een Mobile App **zonder UI (logica)**

Er zijn veel keuzes die je moet maken wanneer je een mobile app gaat ontwikkelen, maar er wordt zelden gesproken of de logica voor de opbouw van de UI in de app hoort ja of nee.

Auteur: Vincent Hendriks

Je zou kunnen denken dat dit een zeer voor de hand liggende keuze is en dit gewoon de verantwoordelijkheid is van de mobile app, maar dat is niet altijd zo.

Traditioneel

Een traditionele opzet is dat er één of meerdere (restful) API endpoints gebouwd worden die de mobile app gebruikt om data op te halen en/of acties uit te voeren. De app gebruikt deze data om de UI op te bouwen. Wat is er mis met deze opzet die al zo vaak gebruikt is? Op zich niets per se, maar wanneer je nu een UI change wilt doorvoeren moet je wel de app bijwerken en opnieuw aanbieden in de app store. En een update betekent dan vaak ook een review. Tegenwoordig willen we zo klein mogelijk iteraties kunnen uitvoeren. We willen kunnen experimenteren met UI's, A/B testen en snel feedback ontvangen. Het wachten op een app store approval is dan niet prettig werken.

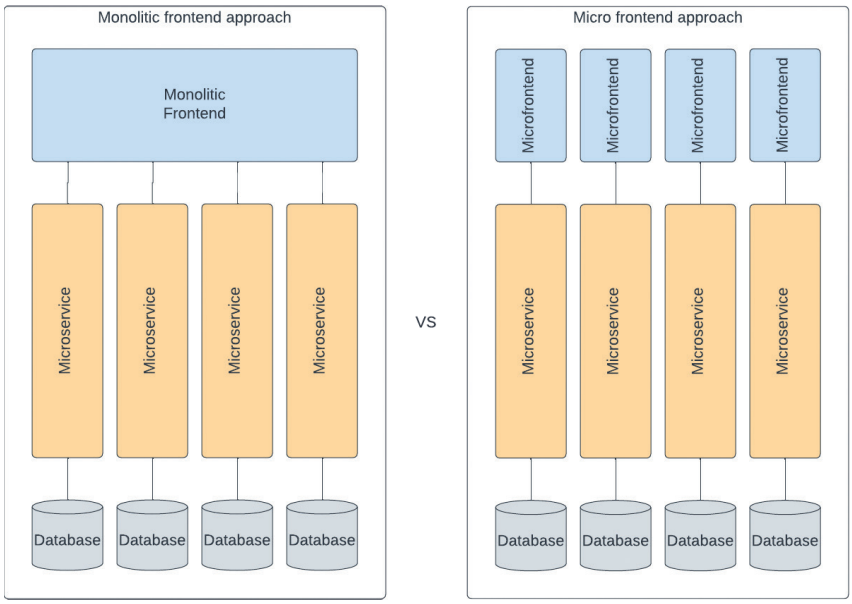
Server Driven UI

De term "Server driven UI" betekent, zoals de naam al doet vermoeden, dat de UI zich laat bepalen door de server.
Bij server driven UI kent de mobile app een reeks aan componenten, maar bepaald de server welke componenten getoond moeten worden. Bij het opstarten begint de app te communiceren met een backend en vraagt simpelweg welke componenten getoond moeten worden. De server geeft een reeks componenten in een vooraf afgesproken format/schema terug en de app toont deze componenten. Deze componenten en opbouw zijn wel allemaal native componenten en daardoor voelt dit ook gewoon als een native ervaring.
Wil je nu de opzet van een pagina aanpassen dan is alleen een aanpassing in de backend nodig, vandaar de term Server Driven UI.
Partijen zoals Airbnb & Lyft staan er bekend om dat ze deze technieken gebruiken.

Je zou kunnen denken dat dit een zeer voor de hand liggende keuze is en dit gewoon de verantwoordelijkheid is van de mobile app, maar dat is niet altijd zo.

Microservices

Microservices in een artikel over hoe je UI opzet in een mobile app? Wanneer microservices, in mijn optiek, echt goed tot zijn recht komen is wanneer de microservices van frontend tot backend helemaal los van elkaar staan. In heel veel situaties wordt dit op backend niveau gedaan, maar als het bij de frontend komt dan is het vaak toch weer 1 geheel wat ze een monolitic frontend noemen. Om volledig los van elkaar staande functionele silo's te ontwikkelen moet je dan ook micro frontends inzetten.
Maar hoe zit dat dan in een mobile app? Als het om het web gaat zijn er meerdere opties mogelijk om micro



frontends te ontwikkelen, maar als het om een mobile app gaat zijn de opties wat beperkt. Er is uiteindelijk echt 1 app die in de app store terecht zal moeten komen. Je zou natuurlijk een module federation aanpak kunnen kiezen waarbij build time er 1 app build uit komt, maar dit betekent dat je voor iedere aanpassing nog steeds een app moet publiceren naar de store met bijbehorende reviews. Bij het gebruik van Server Driven UI heb je in heel veel gevallen geen mobile app update nodig en kun je op de "backend" zoveel aanpassingen doen die nodig zijn zonder ook maar 1 app update. Pas wanneer je de mogelijke componenten wilt uitbreiden of aanpassen zul je een nieuwe app build moeten publiceren naar de store.

Betekent dit dat je per micro frontend/service volledige vrijheid hebt? Nee, je hebt nog wel de afhankelijkheid met wat de mobile app aan mogelijkheden biedt en zul je je naar 1 standaard moeten conformeren. Dit is niet anders bij micro frontends waarbij er in zekere zin gemeenschappelijke afspraken zijn. Het kan tegelijkertijd ook voordelen bieden, want als iedere micro frontend/service zijn eigen stijl hanteert ziet het er nogal snel slordig uit. Server Driven UI helpt dus ook om de huisstijl over je gehele app identiek te krijgen en houden.

Granulariteit

Er zijn verschillende opties die je kunt nemen als het gaat om Server Driven UI. De meest "pure" manier is om op een zo klein mogelijk component toe alles te laten specificeren door de backend, maar je kunt er ook voor kiezen om alleen met wat grotere componenten te werken. Hoe kleiner de componenten die gespecificeerd kunnen worden hoe meer flexibel de backend is en hoe minder vaak er een mobile app update uitgevoerd moet worden. Wel heb je dan een grotere investering nodig om deze techniek te toe te gaan passen t.o.v. wanneer je alleen de wat grotere componenten gebruikt.

In de mobile app die wij, Anycoin Direct, aan het ontwikkelen zijn gebruiken we een vorm van deze techniek ook. Bij de start van het ontwikkelen wisten we al dat wij, net zoals onze web omgeving, een zeer divers aantal scenario's kunnen krijgen die we met enige regelmatig willen aanpassen. Er zit veel logica in onze aankoop/verkoop flow's en om al deze logica in een mobile app te krijgen leek ons niet handig. Dat zou namelijk betekenen dat:

- We al deze processen moesten gaan uitleggen aan een nieuw mobile app development team.
- Aangezien we, na een weloverwogen besluit, voor een native app wilden gaan zou dat betekenen dat we de logica voor zowel android als iOS moesten ontwikkelen en onderhouden.

BIO

Vincent Hendriks

Vincent Hendriks is CTO bij Anycoin Direct en een Software Engineer met meer dan 16 jaar ervaring. Vincent heeft een zeer brede ervaring in onderwerpen zoals C#, .NET/.NET Core, Software architectuur, CQRS, Event Sourcing, SQL, Azure, Docker en nog veel meer.



Uiteindelijk hebben we besloten dat we voor een mix gingen tussen een traditionele opzet en server driven UI. De meeste pagina's gebruiken een traditionele opzet waar de logica volledig in de mobile app zit en er met een (restful) API endpoint de data opgehaald wordt. Maar wanneer je een aankoop wilt doen van een cryptocurrency dan is de backend in control. De app geeft aan dat er een "flow" gestart moet worden en de backend bepaald welk "scherm" getoond moet worden en welke inhoud deze moet hebben. Is er een keuze gemaakt dan gaat de keuze naar de backend en de backend bepaalt het volgende "scherm".
Kijk in dit geval vooral wat het beste past in jouw situatie. We zijn niet allemaal de next Airbnb of Lyft



Server driven UI kan een techniek zijn die het mogelijk maakt om efficiënter, sneller en schaalbaar je mobile app te bouwen.

Backwards compatibility

Wat bij een web omgeving anders is, is dat de gebruiker zelf kiest wanneer hij/zij zijn mobile app update (of niet). Dit betekent dat de backend veelal backward compatible moet zijn met de mobile apps. Natuurlijk kun je hier ook een mechanisme voor introduceren waarbij je een bepaald minimumversie vereist, maar je wilt niet voor iedere app release de gebruikers forceren om eerst te updaten.
Bij een traditionele app betekent dit dat je je api endpoints moet versionen en/of backwards compatible moet houden.
Wanneer je server driven UI gebruikt is dit natuurlijk ook gewoon een aandachtspunt waar je vanaf moment 1 over na moet denken. Je kunt ervoor kiezen om componenten te gaan versioneren en oudere componenten tonen indien nodig. Ook kun je meerdere versies van pagina's ontwikkelen en afhankelijk van de app versie een ander scherm tonen. En je kunt natuurlijk ook gewoon zeggen dat de gebruiker eerst maar eens de app moet bijwerken. It's up to you...

Offline

Wanneer je server driven UI kiest moet je wel goed realiseren dat je app dus beperkt is op momenten dat er geen connectiviteit is. Voor sommige apps is dit helemaal geen probleem, maar voor andere weer wel. Dit is iets wat per app bepaald moet worden. Maar dit is natuurlijk ook een onderwerp waar je goed over na moet denken als je voor de meer traditionele aanpak gaat met api endpoints. In ons geval is er een hele duidelijke scheiding waar de connectiviteit vereist is en waar niet. Voor algemene content zoals welke cryptocurrencies wij aanbieden of de laatst bekende koers cachen we dit in de app. Zou er even geen connectiviteit zijn dan kunnen we de laatst bekende cache tonen. Ga je een flow in waar we Server Driven UI gebruiken dan is dit geen optie. In ons geval geen probleem, want offline een aankoop doen is nou eenmaal niet mogelijk.

Mag dit wel?

Sommige zullen denken dat dit wellicht een issue is waardoor je app wellicht afgekeurd zal worden bij het review proces van google en/of apple. Dit is gelukkig niet het geval, omdat er geen code aangepast wordt. We kunnen bijvoorbeeld geen nieuwe SDK's of libraries gebruiken waarmee we potentieel vervelende dingen zouden kunnen doen die Google en/of Apple niet willen. Wanneer je er over nadenkt is het ook helemaal niet zoveel anders dan dat traditionele apps dit doen. Een traditionele app gebruikt ook dynamische content, waar ook plaatjes getoond zouden kunnen worden die de stores niet zouden willen. Het enige verschil is dat traditioneel de UI logica in de app zit en nu zit de logica buiten de app.

Conclusie

Server driven UI kan een techniek zijn die het mogelijk maakt om efficiënter, sneller en schaalbaar je mobile app te bouwen. Het vergt vaak wel een wat grotere investering in het begin, maar op termijn betekent dit dat je veel minder in je mobile app zelf hoeft te ontwikkelen. Het is een techniek die toegepast kan worden bij native apps in welke taal of framework dan ook.



aws

AWS Is How Now Go Build

Check out AWS Developer Tools



AWS partners with TEQNATION
teqnation.com/aws-partner-page/

Jasper Sprengers

Agile 2: meer dan een upgrade

De zeventien deelnemers hadden vast niet voorzien hoe succesvol hun gezamenlijke brainstorm in februari 2001, samengevat in het Agile Manifesto, zou uitpakken. Voor het gangbare denken over softwareontwikkeling was deze heisessie van ongekende invloed. Scrum zag het licht al in 1995, maar liftte handig mee op het agile succes en is er nu bijna synoniem mee geworden. Veel jonge ontwikkelaars hebben nooit iets anders meegemaakt op de werkvloer.

Maar Agile is ook een wassen neus. Veel organisaties doen maar wat, klagen de coaches, en ze leggen ons in boeken en blogposts uit hoe het wél hoort. Daarentegen mopperen ontwikkelaars ook, maar eerder omdat de adviezen van weleer vastlopen in futloze ceremonies. Wat is er aan de hand? Het Manifesto stond toch voor verstandig afwegen en compromissen maken?

Zeker, maar tegelijk is er ook een lucratieve certificeringsindustrie ontstaan die uit lijfsbehoud vasthoudt aan de letter van de wet. Als Scrum/LeSS/SAFe/Kanban voor jou niet werkt, doe je iets niet goed (wij van wc-eend adviseren wc-eend). Zo lust ik er nog wel één. De medische stand bleef ook eeuwenlang enthousiast aderlaten. Patiënt overleden? Jammer, hebben we nét te weinig adergelaten.

Het is zonde dat de balans in de praktijk is zoekgeraakt. Werkende software boven documentatie? Mooi, source code hoeven we nooit meer te documenteren. De beste architecturen komen uit zelfsturende teams? Mooi, management is alleen maar een lastpost. Toch is die neiging om de extremen op te zoeken niet zo raar als je beseft dat Agile voortkomt uit de Extreme Programming beweging, die – de naam zegt het al – de boel flink wilde opschudden met radicale inzichten. Daar paste geen zorgvuldig wikken en wegen.

Maar voor originele software bestaat geen werkwijze, extreem of gematigd, die voor elk product of team even goed werkt. Vijftien experts bundelden daarom hun kennis en werkten tijdens de pandemie aan een genuanceerde kritiek en grondige herdefinitie van het Agile gedachtengoed in het boek Agile 2 – the next iteration of Agile. De variatie in gender en etniciteit is sowieso al stukken beter in deze groep van niet enkel softwarespecialisten. Het originele Manifesto was hopeloos gekleurd, en dan heb ik het niet over huidskleur. Als je de internationale gemeenschap wil vertellen hoe ze het beste software bouwt, ga dan niet met enkel witte Amerikaanse techneuten bij elkaar zitten. Dan hoor je vooral hoe een elite herenclub in Silicon Valley graag werkt. Uncle Bob Martin zal bij dit woke commentaar wel op zijn achterste benen staan. Stuur het hem vooral door.

Ook Agile 2 heeft waarden en principes, maar deze passen bewust niet op een A4'tje. Daar is de praktijk nu eenmaal te complex voor. Een tipje van de sluier dan: goed leiderschap en individuele verschillen in werkvoorkeur krijgen veel aandacht. En bij alles moet je durven zeggen "it depends". Agile2 doet niet aan one size fits all.

Het is erg welkom dat er een frisse wind waait door het toch wel verstokte denken over Agile, met aandacht voor de mens boven de regels. ●



Jasper Sprengers

SDN

Software Development Network

Word nu lid voor
maar €69,95 per jaar
en ontvang de volgende voordelen!

Wees er snel bij, het eerste jaar dat je lid bent krijg je 50% korting en betaal je maar €35,-!

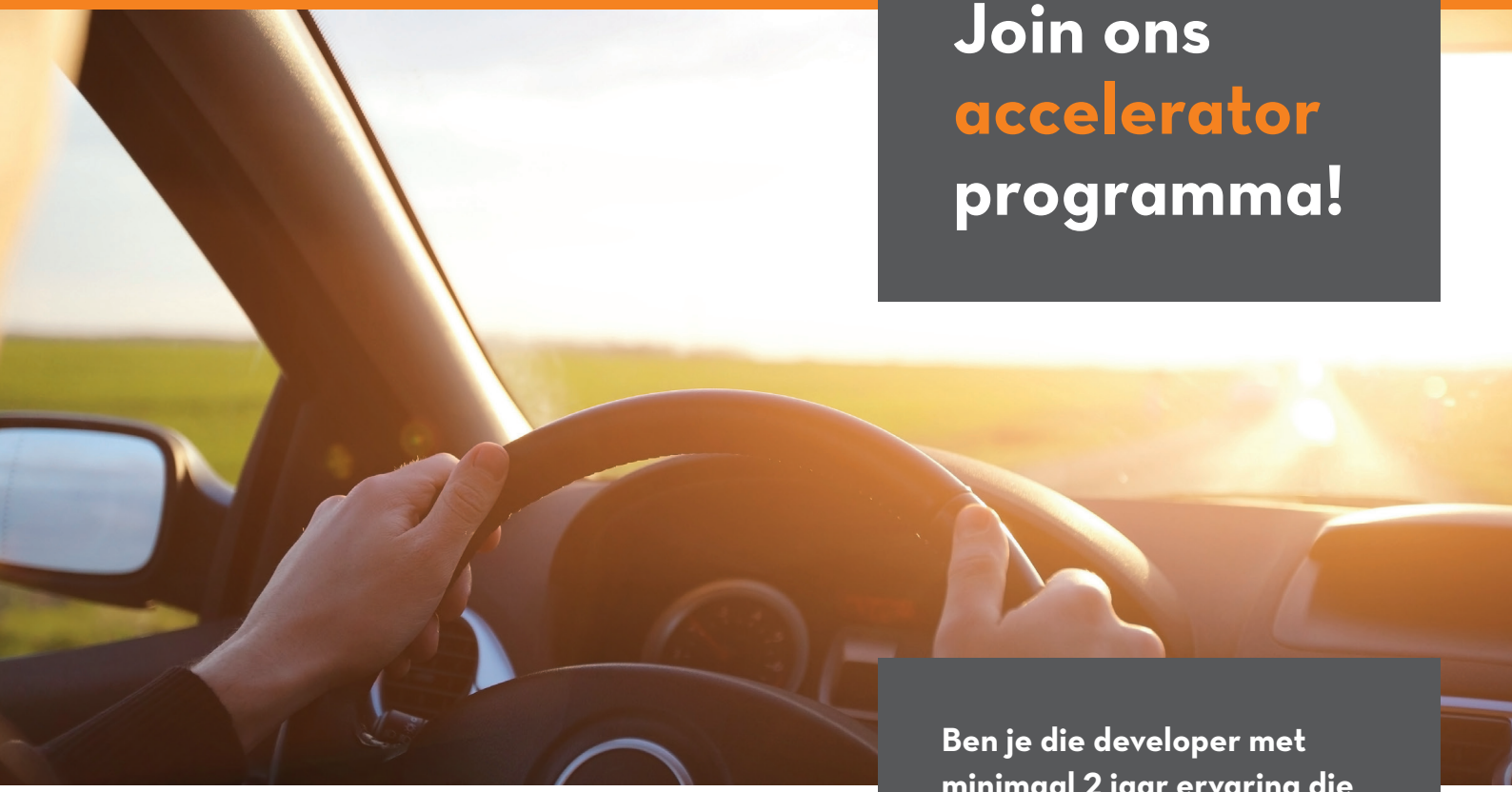
> **Gratis toegang tot Future Tech**
> **Gratis toegang tot SDN University sessions**
> **4 keer per jaar een gedrukt exemplaar van het SDN Magazine**

Voor elke serieuze.NET developer en/of bedrijven die Microsoft technologieën gebruiken is het een must om onderdeel te worden van de SDN. Als je up-to-date wilt blijven, is dit de manier om dat te doen.

Ga naar sdn.nl/lidworden/ voor meer informatie

“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
Ahead of change

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

Ben je die developer met minimaal 2 jaar ervaring die zijn developer kennis naar een next level wil brengen?

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

Kijk voor meer informatie op:
www.ordina.nl/werkenbij

Cloud Days • Security Days • DevOps Days • Testing Days • Frontend Days • Framework Days