SDN MAGAZINE

Nummer
150
apr 2024

SDN Magazine is een uitgave van:
SDN

Van en voor Microsoft & .NET professionals

> De rampzalige illusie van perfecte software

> Experience beats certificates. Every time

> Van Story tot Deployment vanuit P.O. perspectief


Netcompany
Problem solvers
We use the full technology stack to develop smarter, more connected, and sustainable societies.
Join our talented people and create IT solutions that are fit for the future.
Apply today.
www.netcompany.com
Explore here.

Colofon

Uitgave
Software Development Network
27ste jaargang
Nr. 150 • april 2024

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

Auteurs:
Maarten Grootoonk, Stefan van der Wiele, Sander Hoogendoorn, Jeroen Wensen, Rene Elstgeest, Jasper Sprengers, Martijn Broos & Arjen Kraak

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
Lieke Stomps
lstomps@reshift.nl

Beste SDN-ers,

Voor je ligt de tweede editie van dit jaar van het SDN magazine. Met dank aan onze auteurs en de redactiecommissie hebben we wederom een magazine met interessante artikelen. Mocht je feedback of input hebben voor een onderwerp of schrijvers kennen, laat het ons weten via redactie@sdn.nl of @SDN_Community. We nemen graag jullie ideeën en wensen mee.

Future Tech staat voor de deur! De beste nationale en internationale sprekers, zoals keynote spreker James Montemagno zijn te vinden op ons event. Woensdag 17 april 2024 gaat het gebeuren in de Supernova van Jaarbeurs Utrecht. Met de code FutureTech2024-SDN-lid (via FutureTech 2024.eventbrite.nl), krijgen jullie als SDN- leden natuurlijk 85% korting. Graag verwelkomen wij jullie op Future Tech, waar je kunt netwerken met je peers uit de community en toffe exposanten kunt bezoeken op de beursvloer.

De dag voorafgaand aan het event vindt de Masters of .NET plaats, in samenwerking met XPRTZ. Masters of .NET is een roemruchte "funprogging contest", die toegankelijk is voor iedere .NET ontwikkelaar, van junior tot senior. In deze wedstrijd wordt niet de API-kennis, maar de echte programmeervaardigheid getest. Meld je hier aan! (Masters-of-dotNET.eventbrite.nl)

In dit magazine delen Maarten Grootoonk, Stefan van der Wiele, Sander Hoogendoorn, Jeroen Wensen, Rene Elstgeest, Jasper Sprengers, Martijn Broos & Arjen Kraak (weer) hun kennis met jullie!

Veel leesplezier en kom zeker langs bij de SDN-cast op Future Tech!

Implementing ChatGTP within NN

Two years before ChatGPT was launched, NN tested OpenAI's Dutch language model as one of the first companies in Europe. The experiment was successful, and after testing additional AI models, we spent 1.5 years ensuring the safe use of ChatGPT. Now, in 2024, we are one of the leading companies in Europe to have implemented ChatGPT on a large scale for Automated Call Logging. The time we save with reduced post-call logging enables us to improve the support to our customers.

Curious about our ChatGPT journey and challenges? Join our tech talk at TEQnation on May 22nd.

NN

conference.teqnation.com

Interested in working at NN?

nn-careers.com

In dit nummer

8 Tegenwoordig ben ik Product Owner (P.O.) Vroeger niet, toen was ik ontwikkelaar. Soms helpt dat in mijn huidige vak, maar meestal heb ik er nu niets aan, omdat ik me van mijn team niet met de techniek mag bemoeien. Ik wil niet melodramatisch doen, maar als ontwikkelaar had ik het makkelijker. Ik kreeg altijd perfecte specificaties aangeleverd van wat ik moest bouwen en waar het aan moest voldoen. Ik hoefde het alleen maar uit te tikken. Wat een luizenleventje was dat.

15 FUTURE TECH 2024
THE IT CONFERENCE FOR MICROSOFT TECHNOLOGIES
Special

24 When I was nine, I still needed to learn to swim. I was significantly late since I grew up in a village scarred by the flood in February 1953. Luckily, I could take lessons and soon got the hang of it. Not long after, I was the proud owner of an array of swimming diplomas and certificates and even started swimming races.

Ga jij de strijd aan?

De Masters of .NET is een roemruchte "funprogging contest" (gebaseerd op .Net), die toegankelijk is voor iedere .NET ontwikkelaar, van junior tot senior. In deze wedstrijd wordt niet de API-kennis, maar de echte programmeervaardigheid getest. Dit gaat ongetwijfeld voor vuurwerk gaat zorgen!

In de verschillende rondes die de wedstrijd telt, wordt iedere keer een grappige of interessante programmeeropdracht verstrekt, alle teams krijgen dezelfde opdracht die vervolgens binnen een vastgestelde tijd opgelost moet worden. Een opdracht is succesvol uitgevoerd wanneer alle unit tests slagen. Er worden punten toegekend wanneer een oplossing is 'ge-submit' die alle tests laat slagen! Hoe minder tijd je gebruikt, hoe meer punten je krijgt, maar te vaak op submit drukken kost iedere keer punten.

Teams bestaan uit maximaal 2 developers. Tijdens het uitvoeren van de opdracht kan geen gebruik gemaakt worden van andere internet-bronnen dan de Masters of .Net omgeving. Het team dat aan het einde van de dag de meeste punten heeft, mag zich "Master of .Net 2024" noemen. Er zal naast eeuwige roem natuurlijk gestreden worden voor mooie en spectaculaire prijzen!

De spelregels beknopt:

> De wedstrijd bestaat uit meerdere opdrachten, alle teams krijgen steeds tegelijk dezelfde opdracht.
> Het winnende team is dat team dat na afloop van alle opdrachten de meeste punten heeft behaald.
> Voor iedere opdracht is een beperkte hoeveel tijd beschikbaar.
> Een opdracht is succesvol uitgevoerd wanneer.
> Alle unit tests slagen.
> Enkele van die tests zijn niet zichtbaar; er zijn immers altijd hidden requirements, in deze contest zijn ze nog zeer redelijk.
> Met iedere opdracht zijn punten te verdienen. Hierbij geldt:

- Er worden pas punten toegekend wanneer een oplossing is ge-submit die alle tests laat slagen.
- Hoe minder tijd je gebruikt, hoe meer punten je verdient.
- Meer dan éénmaal op submit drukken kost iedere keer punten.
- Tijdens het uitvoeren van de opdracht mag geen gebruik gemaakt worden van andere internet-bronnen dan de Masters of .Net omgeving.

MASTERS OF .NET

SDN

Book your free tickets here

Een SDN Event

Hoofdsponsor: XPRTZ DE PLEK VOOR .NET EXPERTS

16 April 2024

Werkspoorkathedraal, Tractieweg 41 Utrecht

DEVITJOBS.NL

Senior Software Developer

Salarisindicatie:
45.000 - 65.000 EUR

Locatie:
Delft

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

Netcompany

.NET Developer Zorg-ICT

Salarisindicatie:
45.000 - 70.000 EUR

Locatie:
Amsterdam

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

ChipSoft

.NET Developer

Salarisindicatie:
60.000 - 75.000 EUR

Locatie:
Nieuwegein

Technologieën:
.NET MVC, ASP.NET, .NET Framework, Azure, Backend, C#, Cloud, DevOpsvMS-SQL, SQL, Windows Forms, Web, .NET

ORDINA

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

Van Story tot Deployment vanuit P.O. perspectief

Tegenwoordig ben ik Product Owner (P.O.) Vroeger niet, toen was ik ontwikkelaar. Soms helpt dat in mijn huidige vak, maar meestal heb ik er nu niets aan, omdat ik me van mijn team niet met de techniek mag bemoeien. Ik wil niet melodramatisch doen, maar als ontwikkelaar had ik het makkelijker. Ik kreeg altijd perfecte specificaties aangeleverd van wat ik moest bouwen en waar het aan moest voldoen. Ik hoefde het alleen maar uit te tikken. Wat een luizenleventje was dat.

Auteur: Jeroen Wernsen

Maar nu dan, wat een gedoe allemaal om die perfecte specificaties aan te kunnen leveren aan het team. Ik zal eens even uitleggen hoe zwaar ik het heb.

Wat is het probleem?

Als eerste moet ik natuurlijk weten wat het probleem is dat opgelost moet worden. Ik gebruik expres deze woorden, want mijn klanten komen meestal met oplossingen. "Ik wil een lijst kunnen printen van ...". Doorvragen waarom ze dat precies willen is dan de kunst. Soms blijkt dan dat ze echt wel met de juiste oplossing zijn gekomen, maar best vaak blijkt dat hun echte probleem ergens anders begonnen is en dus ook anders (slimmer, goedkoper) opgelost kan worden. Misschien hebben ze die lijst helemaal niet nodig als op strategische plekken op het scherm wat meer informatie wordt toegevoegd. Daarom moet ik dus het achterliggende probleem kennen.

Dat doorvragen heeft nog een ander doel. Ik moet als P.O. begrijpen wat het werk van mijn klant inhoudt, hoe dat werk gedaan wordt en aan welke wetten en regels dat gebonden is. Die Domeinkennis heb ik als ik bij een klant kom nog helemaal niet. Ik weet alleen maar hoe je software moet bouwen. Als ik wat langer bij dezelfde klant zit dan komt die Domeinkennis enigszins, maar mijn klant blijft een zeer belangrijke partner voor het verstrekken van die kennis.

Hoe willen we dit oplossen?

Als ik het probleem ken, dan kan ik gaan werken aan een oplossing. Die oplossing bedenk ik natuurlijk niet zelf, ik moet al zoveel doen. Nee, ik maak een zogenaamde "User Story" en bespreek dat met het team. Voor de leken: Een User Story beschrijft op een bepaalde manier wat het probleem is en wanneer we dat probleem als opgelost beschouwen. Bij complexe problemen gaat hier veel werk in zitten, dan moet ik diagrammetjes en dergelijke maken om het probleem duidelijk te kunnen uitleggen. De meeste problemen worden ook niet met slechts 1 User Story opgelost. In die gevallen wordt een serie van gerelateerde User Stories gemaakt, die elk een deel van het probleem oplossen.

Het team en ik bespreken dus tijdens een zogenaamde "Refinement sessie" de User Story om samen te bedenken wat we gaan opleveren om het probleem te tackelen. Dit vind ik een van de leukste onderdelen van mijn vak: het sparren met het team hoe we iets slim oplossen. Hier merk je ook de grootste verschillen tussen teams: sommige teams willen gewoon van mij horen wat ze moeten doen en sommige teams denken actief mee met het vinden van de beste oplossing. Je snapt dat ik dat laatste het liefste heb. Dat is minder werk voor mij.

Hoe bewijzen we dat het opgelost is?

Ik geloof het ontwikkelteam natuurlijk niet zomaar op hun mooie blauwe (of bruine) ogen als ze zeggen dat het opgelost is. Ze moeten dat aan mij, en de klant, kunnen bewijzen aan de hand van zogenaamde "acceptatiecriteria" die ik bij de voorbereiding al aan de User Story had toegevoegd. Tijdens een live demo door het team checken de klant en ik of aan alle acceptatiecriteria voldaan is. Als dat zo is, dan is het probleem opgelost. Als dat niet zo is dan is er nog werk aan de winkel voor het team.

Die acceptatiecriteria zijn best belangrijk, want die geven voor een groot deel de scope van zo'n oplossing aan en vaak gaan ze niet alleen over het probleem, maar ook over randvoorwaarden zoals beveiliging of gebruikerservaring, dat soort zaken.

Hou het team in toom!

Ontwikkelaars zijn kenniswerkers, vaak halen ze hun voldoening uit het ontdekken van nieuwe frameworks, technieken of toepassingen. Het is belangrijk voor hen om bij te blijven in hun vak, maar soms ontaard dat in een oplossing die te ver is doorontwikkeld of veel te ingewikkeld is gemaakt. Daarom specificeer ik in een User Story ook wanneer het goed genoeg is. Dat doe ik m.b.v. de "Out of scope" specificaties die aan de ontwikkelaars duidelijk maken wat ik te ver vind gaan. Zo hoop ik te voorkomen dat mijn probleem opgelost zou zijn met een fiets, maar een auto geleverd krijg. Dat doe ik niet om het team te pesten, maar om te zorgen dat de tijd van de ontwikkelaars zinnig besteed wordt. Goed is goed genoeg, perfect is zonde van de tijd.

Wendbaarheid gevraagd

Tot nu toe lijkt het alsof ik een luizenleventje heb. Maar de klussen waarvoor ik ingehuurd word gaan natuurlijk niet over simpele problemen. Het zijn altijd complexe, grote vraagstukken waar een klant zelf niet meer uitkomt, of geen capaciteit voor heeft. En die los je niet op met een paar User Stories. Om dit soort oplossingen te bouwen moet ik, in samenwerking met de klant en het team, het probleem opdelen in grote brokken (Epics) die weer onderverdeeld zijn in gerelateerde onderdelen (Features). De Features vallen dan weer uiteen in verschillende User Stories. Het kost veel tijd en doorvragen om dat allemaal inzichtelijk te maken, en ondertussen willen de ontwikkelaars wel aan de slag. Dat lossen we op door "Agile" te werken: we beginnen met wat we weten en terwijl ik en het team steeds meer kennis opdoen, werken we toe naar de uiteindelijke oplossing.

Soms blijkt dat we eerder in de ontwikkeling iets gebouwd hebben wat niet afdoende is. Dan nemen we de tijd om het te verbeteren totdat het goed genoeg is. Soms hadden we een onderdeel wel goed genoeg gebouwd, maar is de klant van mening veranderd. Ook dan kunnen we het veranderen. Geen probleem, de klant is immers Koning. Belangrijk is wel dat ik dan aan de klant duidelijk maak dat elk nieuw inzicht ook extra werk, dus extra kosten betekent.

"Is het al af?"

Werkelijk elke klant wil weten wanneer het af is. Maar het antwoord op die vraag is niet zo simpel bij het soort grote klussen welke ik doe. In het begin van het traject is er nog heel veel onduidelijk, dus zijn de schattingen van de benodigde hoeveelheid werk ook onnauwkeurig. Maar hoe meer er duidelijk wordt, hoe beter deze schattingen worden. Die schattingen maak ik natuurlijk niet, die laat ik het team opstellen op basis van de deelproblemen die ik ondertussen boven water heb weten te krijgen.

Ik ben wel degene die aan de klant moet vertellen wat de totale schatting is en hoe ver we denken dat we zijn. Dat vereist altijd tact, iets wat heb ik niet in ruime mate bezit. Dit faken maakt mijn werk echt zwaar. Af en toe een schouderklopje zou best welkom zijn.

Het is vrij gebruikelijk dat de schattingen ruimer zijn dan het budget. In die gevallen wordt in overleg met de klant bepaald wat de minimum requirements zijn om een basis versie (MVP) van de oplossing van het probleem op te leveren. De Epics, Features en User Stories die toewerken naar die MVP krijgen dan bij de planning van alles prioriteit. De onderdelen die niet-MVP zijn, komen dan aan het einde, als er nog tijd en budget over is. En anders maar niet.

Soms gaat het om serieus groot probleem, waar meerdere teams tegelijkertijd aan werken. Dan komt er nog een heel niveau van overleg bij: afstemming tussen de teams onderling. Op zich is dat wel leuk, want dan kan je overleggen met collega P.O.'s die snappen waarom je het zo zwaar hebt. Maar ja, die afstemming maakt het natuurlijk nog wat zwaarder. Van de regen in de drup dus.

Ten slotte

Tot nu toe zijn mijn klanten altijd blij geweest met mijn inzet en het opgeleverde resultaat. Uiteraard claim ik alle verantwoordelijkheid voor succesvolle oplossingen. Mocht er in de toekomst een klant niet tevreden zijn met het resultaat, dan kan dat alleen maar aan de ontwikkelaars liggen.
Je vraagt je misschien af waarom ik in hemelsnaam dit werk doe, als het allemaal zo moeilijk is. Maar dat is precies ook het antwoord: hoe groter de moeilijkheidsfactor, hoe groter de bevrediging als het allemaal tot een goed resultaat leidt. Tel daarbij op het sociale aspect van het overleggen met de klant, het team, solution architecten, beheerders, customer support en de overige P.O.'s. Ik leer allerlei verschillende leuke mensen kennen, en ik weet steeds een beetje meer af van hoe zaken werken in deze wereld.
Je kan het vergelijken met het krijgen van kinderen: je zit er de rest van je leven aan vast maar je krijgt er zoveel voor terug.
Zonder wrijving geen glans.
Zonder dalen geen toppen.

Enfin, je snapt het wel. O... en ik heb in dit artikel misschien een paar keer de draak gestoken met de rol van product owner. Het is een prachtig vak waar je zo links en rechts wel de nodige vooroordelen van terug hoort komen 😊.

Decentrale identiteiten: een nieuwe manier om vertrouwen te creëren in het digitale domein

Digitale identiteit is een essentieel onderdeel van onze online interacties. Het stelt ons in staat om onszelf te authenticeren, te autoriseren en te verifiëren bij verschillende diensten en platforms. Maar hoe kunnen we er zeker van zijn dat onze digitale identiteit veilig, betrouwbaar en privacyvriendelijk is? In dit artikel bespreken we het concept van decentrale identiteit, een nieuwe benadering die gebaseerd is op open standaarden en die gebruikers meer controle en gemak biedt over hun eigen identiteit. We vergelijken decentrale identiteit met de traditionele centrale identiteit, en laten zien hoe het gebruik van decentralised identifiers (DID) en verifiable credentials (VC) kan leiden tot een transparanter, veiliger en inclusiever identiteitssysteem. We sluiten af met een voorbeeld van hoe Microsoft Entra Verified ID deze technieken implementeert en gebruikt om decentrale identiteiten te faciliteren.

Auteur: Stefan van der Wiele

Centrale identiteit versus decentrale identiteit

Laten we beginnen met een voorbeeld van twee bedrijven die data willen uitwisselen, maar geen vertrouwensband hebben. Hoe kunnen ze elkaars identiteit verifiëren en zeker weten dat de data die ze ontvangen geldig en authentiek is? Een veelgebruikte oplossing is om gebruik te maken van een centrale identiteit, waarbij beide partijen vertrouwen op een derde partij, de identity provider (IdP), die de identiteit van de gebruikers beheert en bevestigt. Voorbeelden van technieken die gebaseerd zijn op centrale identiteit zijn OpenID Connect en SAML, die vaak gebruikt worden voor single sign-on (SSO) en federated identity. Deze technieken bieden een hoog gebruiksgemak, maar hebben ook een aantal nadelen, zoals:

- Afhankelijkheid: de gebruikers zijn afhankelijk van de IdP voor het beheren en beschermen van hun identiteit en data. Als de IdP offline gaat, gehackt wordt of zijn beleid verandert, kunnen de gebruikers hun toegang tot de diensten verliezen of hun privacy in gevaar brengen.
- Centralisatie: de IdP heeft een centrale rol in het identiteitssysteem, en kan daardoor een machtspositie verwerven en misbruiken. De IdP kan ook een doelwit worden voor aanvallers die de identiteit en data van de gebruikers willen stelen of manipuleren.
- Fragmentatie: de gebruikers moeten meerdere accounts en wachtwoorden beheren voor verschillende diensten en IdP's, wat leidt tot een versnipperde en inconsistente identiteit. De gebruikers hebben ook weinig controle over welke data ze delen met de diensten en de IdP's, en kunnen niet gemakkelijk hun toestemming intrekken of wijzigen.

Een alternatieve oplossing is om gebruik te maken van een decentrale identiteit, waarbij de gebruikers zelf eigenaar zijn van hun identiteit en data, en deze opslaan in een persoonlijke wallet. De gebruikers kunnen hun identiteit en data delen met de partijen die ze vertrouwen, zonder tussenkomst van een centrale autoriteit. De vertrouwensband tussen de partijen is gebaseerd op een decentrale infrastructuur, zoals een

blockchain of een distributed ledger, die de integriteit en authenticiteit van de identiteit en data garandeert. Voorbeelden van technieken die gebaseerd zijn op decentrale identiteit zijn decentralised identifiers (DID) en verifiable credentials (VC), die we in de volgende secties nader zullen toelichten. Deze technieken bieden een aantal voordelen, zoals:

- Onafhankelijkheid: de gebruikers zijn niet afhankelijk van een derde partij voor het beheren en beschermen van hun identiteit en data. Ze kunnen zelf kiezen met wie ze hun identiteit en data delen, en hoe ze deze opslaan en beveiligen.
- Decentralisatie: er is geen centrale autoriteit die het identiteitssysteem domineert of compromitteert. De identiteit en data van de gebruikers zijn verspreid over een netwerk van nodes, die samenwerken om de identiteit en data te valideren en te synchroniseren.
- Consistentie: de gebruikers hebben één consistente en draagbare identiteit, die ze kunnen gebruiken voor verschillende diensten en platforms. De gebruikers hebben ook meer controle over welke data ze delen, en kunnen hun toestemming gemakkelijk verlenen, intrekken of wijzigen.

Decentralised identifiers (DID)

Een decentralised identifier (DID) is een unieke, persistente en resoluteerbare identifier die verwijst naar een entiteit, zoals een persoon, een organisatie of een ding. Een DID bestaat uit een URI die begint met did: gevolgd door een specifieke methode en een uniek identificatienummer. Bijvoorbeeld: did:web:example.com. Een DID kan worden opgeslagen op een decentrale infrastructuur, zoals een blockchain of een distributed ledger, die de eigendom en de controle over de DID garandeert.

Een DID kan ook worden gekoppeld aan een DID-document, dat meer informatie bevat over de entiteit, zoals de publieke sleutels, de service endpoints en de authenticatiemethoden. Een DID-document kan worden opgevraagd door de DID te resolven via een DID-resolver, die de juiste methode gebruikt om het DID-document te vinden en te retourneren.

Er zijn verschillende DID-methoden, die verschillen in de manier waarop ze de DID's creëren, opslaan, resolven en beheren. Een voorbeeld van een DID-methode is did:web, die gebruik maakt van het bestaande webdomeinnaamsysteem (DNS) om de DID's te hosten en te resolven. Een did:web DID ziet eruit als een normale URL, maar begint met `did:web:` in plaats van https://. Bijvoorbeeld: did:web:example.com. Een did:web DID-document kan worden opgevraagd door de DID te vervangen

door https:// en .well-known/did.json toe te voegen aan het einde. Bijvoorbeeld: https://example.com/.well-known/did.json. Een did:web DID-document kan worden opgeslagen op een webserver die verbonden is met het DNS, en kan worden beveiligd met HTTPS en DNSSEC.

Verifiable credentials (VC)

Een verifiable credential (VC) is een digitaal bewijsstuk dat een bepaalde claim of eigenschap bevestigt over een entiteit, zoals een naam, een leeftijd of een diploma. Een VC bestaat uit een gestructureerd gegevensformaat, zoals JSON of JSON-LD, dat de claim, de issuer, de holder en de verificatiegegevens bevat. Een VC kan worden ondertekend door de issuer met een digitale handtekening, die de authenticiteit en de integriteit van de VC garandeert. Een VC kan ook worden verpakt in een verifiable presentation (VP), die een of meer VC's bevat die de holder wil presenteren aan een verifier. Een VP kan ook worden ondertekend door de holder met een digitale handtekening, die de toestemming en de betrokkenheid van de holder bevestigt.

Er zijn drie belangrijke rollen in het VC-ecosysteem: de issuer, de holder en de verifier. De issuer is de entiteit die de VC uitgeeft aan de holder, nadat deze de claim heeft gevalideerd. De issuer kan bijvoorbeeld een overheidsinstantie, een onderwijsinstelling of een bedrijf zijn. De holder is de entiteit die de VC ontvangt en opslaat in een persoonlijke wallet. De holder kan bijvoorbeeld een persoon, een organisatie of een ding zijn. De verifier is de entiteit die de VC controleert en verifieert, om te bepalen of de claim geldig en betrouwbaar is. De verifier kan bijvoorbeeld een bank, een webshop of een luchthaven zijn.

Laten we een voorbeeld bekijken van hoe VC's kunnen worden gebruikt in een praktische situatie. Stel dat Alice een online aankoop wil doen bij Bob's webshop, maar dat ze daarvoor haar leeftijd moet bewijzen. Alice heeft een VC die bewijst dat ze 18 jaar of ouder is, en die ze heeft ontvangen van een betrouwbare issuer, bijvoorbeeld de gemeente. Alice kan deze VC opslaan in haar persoonlijke wallet, die de rol van holder speelt. Alice kan vervolgens de VC presenteren aan Bob's webshop, die de rol van verifier speelt. Bob's webshop kan de VC controleren en verifiëren, om te bepalen of Alice aan de leeftijdseis voldoet. Bob's webshop hoeft niet te weten wie Alice precies is, of welke andere gegevens ze heeft. Alice kan haar privacy beschermen door alleen de relevante informatie te delen.

Om de VC te presenteren, moet Alice een verifiable presentation (VP) genereren, die de VC bevat die ze wil delen met Bob's webshop. Alice kan ook een digitale handtekening zetten op de VP, om te bewijzen dat ze de rechtmatige eigenaar is van de VC. Alice kan de VP vervolgens versturen naar Bob's webshop, via een beveiligd kanaal. Bob's webshop kan de VP ontvangen en de volgende stappen uitvoeren om de VC te verifiëren:

- De digitale handtekening van Alice op de VP controleren, om te bevestigen dat Alice de holder is van de VC.
- De digitale handtekening van de issuer op de VC controleren, om te bevestigen dat de VC authentiek en ongewijzigd is.
- De status van de VC controleren, om te bevestigen dat de VC niet is ingetrokken of verlopen.
- De vertrouwenswaardigheid van de issuer controleren, om te bevestigen dat de issuer een bevoegde en betrouwbare partij is.
- De inhoud van de VC controleren, om te bevestigen dat de claim over Alice geldig en relevant is.

Als al deze stappen succesvol zijn, kan Bob's webshop de VC accepteren en Alice toegang verlenen tot de online aankoop. Bob's webshop kan ook een ontvangstbewijs sturen naar Alice, dat ook een VC kan zijn, om de transactie te bevestigen.

Decentrale identiteiten en verifiable credentials

Zoals we hebben gezien, kunnen verifiable credentials een krachtig middel zijn om vertrouwen te creëren in het digitale domein, zonder afhankelijk te zijn van een centrale autoriteit. Maar hoe kunnen we verifiable credentials koppelen aan decentrale identiteiten, die gebaseerd zijn op decentralised identifiers (DID)? Het antwoord is dat we DID's kunnen gebruiken om de identiteit en de publieke sleutels van de issuers, de holders en de verifiers te identificeren en te resolven. Dit maakt het mogelijk om de digitale handtekeningen te verifiëren, de service endpoints te

Verifiable credentials kunnen een krachtig middel zijn om vertrouwen te creëren in het digitale domein.

vinden en de authenticatiemethoden te ondersteunen. Bovendien kunnen we DID's gebruiken om de subjecten van de claims te identificeren, zonder hun privacy te schenden.

Laten we teruggaan naar ons voorbeeld van Alice en Bob's webshop, en zien hoe DID's het proces kunnen verbeteren. Stel dat Alice een DID heeft, bijvoorbeeld did:web:alice.example.com, die verwijst naar haar identiteit en haar publieke sleutel. Alice kan deze DID gebruiken om haar VC te koppelen aan haar identiteit, door de DID op te nemen in het `subject` veld van de VC. Alice kan ook haar DID gebruiken om haar VP te ondertekenen, door de DID op te nemen in het verificationMethod veld van de VP. Alice kan vervolgens haar VP versturen naar Bob's webshop, die de DID van Alice kan resolven om haar publieke sleutel te verkrijgen en haar handtekening te controleren.

Bob's webshop kan ook een DID hebben, bijvoorbeeld did:web:bob.example.com, die verwijst naar zijn identiteit en zijn publieke sleutel. Bob's webshop kan deze DID gebruiken om zijn ontvangstbewijs te ondertekenen, door de DID op te nemen in het verificationMethod veld van de VC. Bob's webshop kan ook zijn DID gebruiken om zijn service endpoint te publiceren, door de DID op te nemen in het `service` veld van zijn DID-document. Alice kan de DID van Bob's webshop resolven om zijn publieke sleutel te verkrijgen en zijn handtekening te controleren, en om zijn service endpoint te vinden en zijn ontvangstbewijs te ontvangen.

De issuer van de VC van Alice kan ook een DID hebben, bijvoorbeeld did:web:gemeente.example.com, die verwijst naar zijn identiteit en zijn publieke sleutel. De issuer kan deze DID gebruiken om zijn VC te ondertekenen, door de DID op te nemen in het verificationMethod veld van de VC. De issuer kan ook zijn DID gebruiken om zijn statusinformatie te publiceren, door de DID op te nemen in het service veld van zijn DID-document. Bob's webshop kan de DID van de issuer resolven om zijn publieke sleutel te verkrijgen en zijn handtekening te controleren, en om zijn statusinformatie te vinden en de geldigheid van de VC te controleren.

Door DID's te gebruiken, kunnen we dus een decentraal identiteitssysteem opbouwen, dat verifiable credentials ondersteunt en versterkt. Dit systeem biedt de volgende voordelen:

- Flexibiliteit: de gebruikers kunnen verschillende DID-methoden kiezen, afhankelijk van hun behoeften en voorkeuren. Ze kunnen ook meerdere DID's hebben, voor verschillende doeleinden en contexten.
- Interoperabiliteit: de partijen kunnen gemakkelijk communiceren en samenwerken, door gebruik te maken van open standaarden en protocollen. Ze kunnen ook elkaars DID's resolven, ongeacht de onderliggende infrastructuur of technologie.
- Zelfsoevereiniteit: de gebruikers hebben volledige controle en eigenaarschap over hun identiteit en data. Ze kunnen zelf beslissen welke DID's ze gebruiken, welke VC's ze delen, en met wie ze vertrouwen.

Microsoft Entra Verified ID en decentrale identiteiten

Microsoft Entra Verified ID is een oplossing die decentrale identiteiten en verifiable credentials mogelijk maakt, door gebruik te maken van Microsoft Azure en Microsoft Entra. Microsoft Entra Verified ID biedt de volgende functies en voordelen:

- Een persoonlijke wallet voor de gebruikers, waar ze hun VC's kunnen beheren en gebruiken. De wallet is ingebouw in de Microsoft Entra Authenticator app
- Een verificatieplatform voor de verifiers, waar ze de VC's van de gebruikers kunnen controleren en verifiëren.
- Een uitgifteplatform voor de issuers, waar ze VC's kunnen creëren en uitgeven aan de gebruikers. Het platform maakt gebruik van de W3C-standaarden voor VC's en de decentralized ID standaard en verschillende DID-methods.
- Een ecosysteem van partners, die VC's kunnen aanbieden of accepteren voor verschillende doeleinden en sectoren. Het ecosysteem omvat onder andere onderwijsinstellingen, overheidsinstanties, financiële instellingen en gezondheidszorgorganisaties.

Conclusie

In dit artikel hebben we het concept van decentrale identiteit besproken, een nieuwe benadering die gebaseerd is op open standaarden en die gebruikers meer controle en gemak biedt over hun eigen identiteit. We hebben het verschil tussen decentrale identiteit en de traditionele centrale identiteit vergeleken, en de voordelen van de decentrale identiteit benadrukt. We hebben ook de technieken van decentralised identifiers (DID) en verifiable credentials (VC) toegelicht, en laten zien hoe ze samenwerken om een transparant, veilig en inclusief identiteitssysteem te creëren. We hebben afgesloten met een voorbeeld van hoe Microsoft Entra Verified ID deze technieken implementeert en gebruikt om decentrale identiteiten te faciliteren.

We hopen dat dit artikel je een beter inzicht heeft gegeven in de mogelijkheden en uitdagingen van decentrale identiteit, en dat je geïnspireerd bent om zelf te experimenteren met deze innovatieve technologie. Decentrale identiteit heeft het potentieel om de digitale samenleving te transformeren, door meer vertrouwen, privacy en zelfsoevereiniteit te bieden aan de gebruikers.

Come see us at Future Tech!

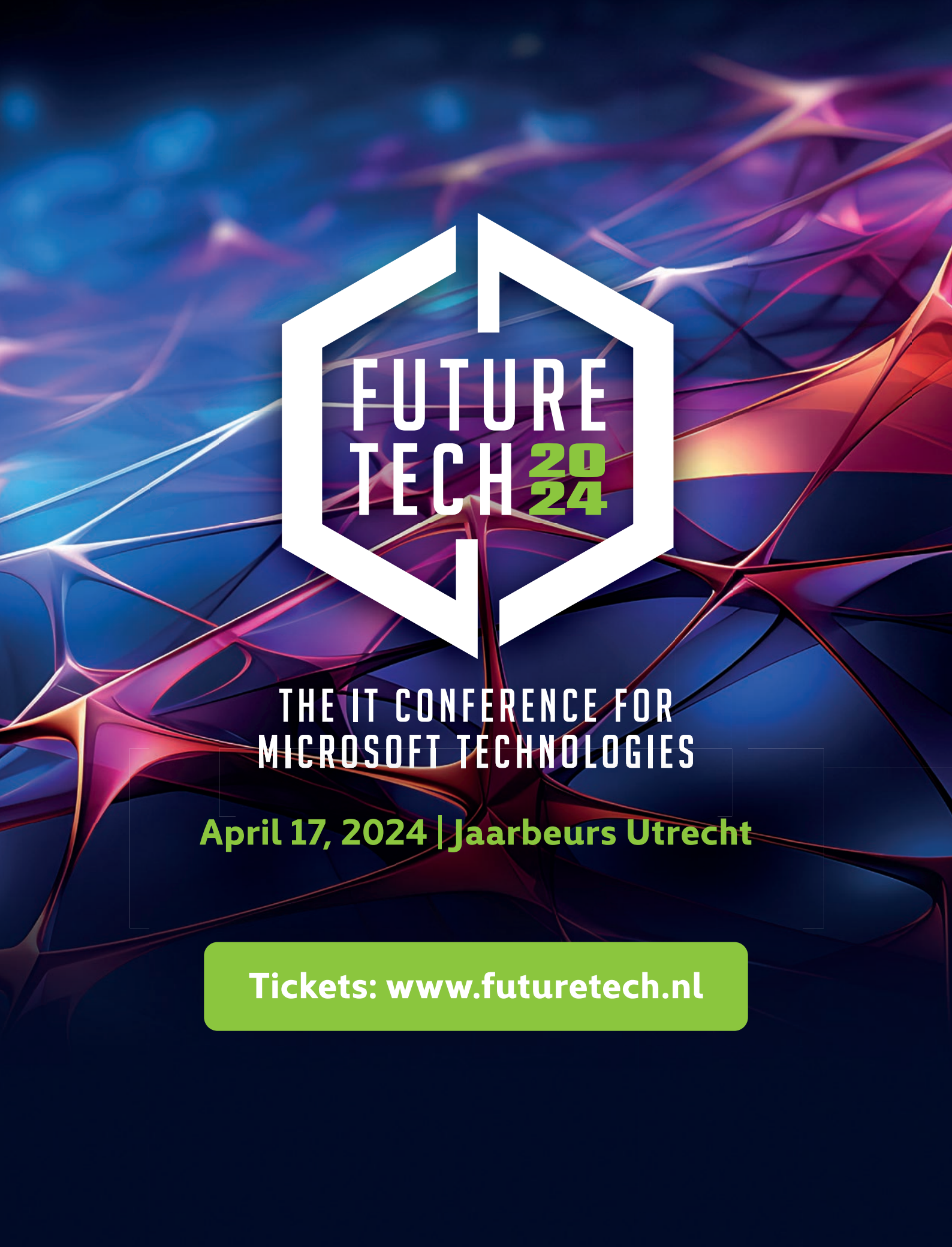
Stop by our stand, meet our colleagues, and chat about various topics. Whether you're intrigued by our projects, curious about our methodology, or simply want to discuss what inspires you, we look forward to talking with you. Plus, don't miss the chance to participate in our giveaway! We're excited to meet you!

De beste bedrijfs-kritische software in Azure

Betabit is een echt familiebedrijf dat in 2002 is opgericht door de broers Olav en Luc Gimbrère.

Als toegewijd Microsoft Solutions Partner, met meer dan 15 jaar Azure ervaring, zijn wij martkleider in softwareontwikkeling, -duiding en opleiding in Microsoft Azure. Van Cloud Native development tot beheer en onderhoud, van software (security) assessments tot gepersonaliseerde trainingen.

Betabit vormt samen met YieldDD en Virtual Vaults een groep met een unieke mix van expertise en ervaring.


FUTURE
TECH 2024
THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES
April 17, 2024 | Jaarbeurs Utrecht
Tickets: www.futuretech.nl

Session ABN AMRO

The future of Cobol and mainframe at ABN AMRO

Dennis Kornet

Learn how ABN AMRO envisions processing 3 billion payments per year, at peak loads of 8000 transactions in the future, using the power of the mainframe.

ABN AMRO's strategy of retaining proven technology, introducing alternatives like Java on Mainframe and using of AI where possible during this transition (and where not).

Session Netcompany

Microservices message processing system – practical case overview

Szymon Jankowski

During the presentation, I will showcase the process of introducing design changes to the microservices message processing system. To support that, we will be looking at a practical case we had in Netcompany. We will discuss the drivers of original design ideas, the discovery of new requirements, and changes they incurred on the original design, as well as whether it supported quick adjustment's introduction. We will be talking about features of the message broker - RabbitMQ and MassTransit library that were helping and disturbing us during the development of the system. This presentation is a practical case overview of service communication and how requirements changes can affect it.

Impact maken in de zorg-ICT?

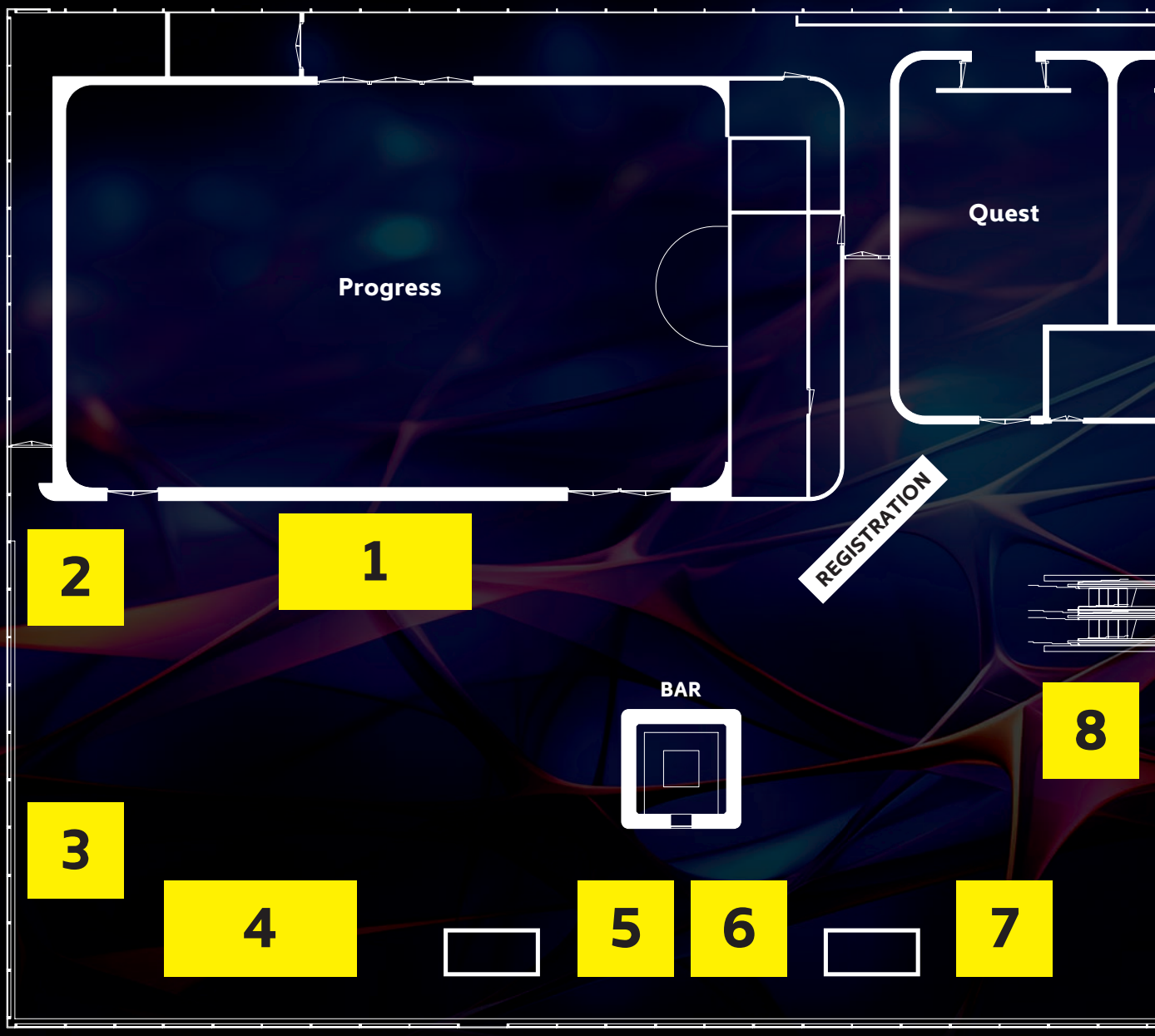
Kom werken bij ChipSoft!

werkenbijchipsoft.nl

Bekijk hier onze vacatures!

ChipSoft

Floorplan Future Tech 2024



Netcompany	1	Betabit	5
SDN Cast	2	Ordina	6
Chipsoft	3	Luminis	7
	4	ABN AMRO	8

*Subject to change


Expedition
Mission 1
Mission 2
Community &
Startup plein
lift
Workshops

Timetable Future Tech 2024

	Progress	Mission 1	Mission 2
10:00 am	**PROGRESS** 10:00 am > 10 min Official opening Future Tech 2024		
	PROGRESS 10:10 am > 30 min Keynote: The AI, Cloud-Native, Run Anywhere Future of .NET **James Montemagno**		
10:40 am	10:40 am > 20 min Coffee Break		
11:00 am	11:00 am > 45 min Microservices message processing system – practical case overview **Szymon Jankowski**	11:00 am > 45 min 1,2,3 ... testing : is this thing on(line)? Meet your new Microsoft Testing tools. **Mike Martin**	11:00 am > 45 min Lessons learned from analyzing enormous time data streams with Azure Data Explorer **Toon Vanhoutte**
12:00 pm	12:00 pm > 45 min How to enhance your own AI services using Semantic Kernel **Håkan Silfvernagel**	12:00 pm > 45 min Shipping features without the angry emails **Maarten Grootoonk**	**Lightning talks** 12:00 pm > 15 min The 3 Ws of Sustainable IT and Green Softw **Saravanan K Nagarajan** 12:15 pm > 15 min Hot compatibility: the new kid on the block **Jarne Van Aerde** 12:30 pm > 15 min Accessibility powered by AI **Ramona Domen**
	12:45 pm > 60 min Lunch Break		
1:45 pm	**PROGRESS** 1:45 pm > 30 min Keynote: The art of getting 💩 done **Henry Been**		
2:30 pm	2:30 pm > 45 min Reviewing NuGet Packages security easily using OpenSSF Scorecard **Niels Tanis**	2:30 pm > 45 min AI in HealthCare Practice by Marc Somberg & Patrick Antonissen **Marc Somberg & Patrick Antonissen**	2:30 pm > 45 min Say goodbye to building boring APIs with Azure Data API Builder **Sander ten Brinke**
3:15 pm	3:15 pm > 30 min Coffee Break		
3:45 pm	3:45 pm > 45 min Living in your own bubble: Introduce DDD into legacy software **Jacob Duijzer**	3:45 pm > 45 min Microservices Orchestration with Azure Container Apps **Eduard Keilholz**	3:45 pm > 45 min Cross the streams for Microsoft Fabric Real-Time Analytics **Sander van de Velde**
4:45 pm	4:45 pm > 45 min Building Future-Ready Apps with .NET 8 and Azure Serverless Ecosystem **Stas Lebedenko**	4:45 pm > 45 min Enabling Efficiency in Continuous Delivery, Test Automation and SRE: Insights from ASMLs Team in Hig **Andrei Kniazev**	4:45 pm > 45 min Lights, camera, action! Building distributed applications with Dapr Actors **Marc Duiker**
5:30 pm	5:30 pm > 30 min Network & Drinks		

*Subject to change

Wednesday, April 17, 2024

	Expedition	Quest	Workshop Area
	11:00 am > 45 min The future of Cobol and mainframe at ABN AMRO **Dennis Kornet**	11:00 am > 45 min Build your own Copilot with Azure AI Studio **Henk Boelman**	11:00 am > 105 min Playing with domain models in code **Nico Krijnen & Niels Marsman**
	12:00 pm > 45 min GitHub Copilot Beyond the Basics - 10 Ways to Elevate Your Coding **Thijs Limmen & Yuliya Khadasevich**	12:00 pm > 45 min TBA **Session by Gerald Versluis**	
	2:30 pm > 45 min Build Your Own Star Wars Droid **Goran Vuksic**	2:30 pm > 45 min Developer productivity shouldn't breach security **Bart Lannoeye**	2:30 pm - 5:30 pm GitHub Copilot: The Smartest Way to Code **Lukas Durovsky**
	3:45 pm > 45 min Building Future-Ready Apps with .NET 8 and Azure Serverless Ecosystem **Stas Lebedenko**	3:45 pm > 45 min DAPR and .NET Aspire: A royal wedding **Florian van Dillen**	
	4:45 pm > 45 min A fun and absurd introduction to Vector Databases **Alexander Chatzizacharias**	4:45 pm > 45 min How to sell a big refactor or rewrite to the business? **Ivett Ördög**	

Session ChipSoft

AI in HealthCare Practice

Marc Somberg & Patrick Antonissen

AI has unprecedented possibilities, of which we are only just at the beginning. Particularly in healthcare with its enormous amounts of different data, where different conclusions can be drawn and where medical knowledge never stands still, AI can offer a nice addition. In this session we will discuss the problems and possible solutions that we have encountered as a leading supplier of Health Information Systems. The actual integration is also discussed. Our solution offers the possibility to feed external digital services with specific patient data, results are obtained and presented to the end user. aspects such as privacy protection and cloud solutions are also discussed.

Session Ordina

Shipping features without the angry emails

Maarten Grootoonk

Features get added to applications all the time, as developers we try our best to make an educated guess on how the features will perform in production and how users will react to the changes. Guessing wrong can result in angry emails and unwanted calls.

Join us in this session to learn how feature flags can be used to ship features and changes without the angry emails. We will cover implementing and fine-tuning feature flags, discuss various approaches, and take a deep dive into the tools that Azure offers.

FUTURE TECH 2024

THE IT CONFERENCE FOR MICROSOFT TECHNOLOGIES

Future Tech thanks her partners

Co-sponsors

ABN·AMRO

Netcompany

Partners

betabit
samen bijzonder maken;

ChipSoft

luminis
Conversing worlds

ORDINA

Experience beats certificates **Every time**

When I was nine, I still needed to learn to swim. I was significantly late since I grew up in a village scarred by the flood in February 1953. Luckily, I could take lessons and soon got the hang of it. Not long after, I was the proud owner of an array of swimming diplomas and certificates and even started swimming races.

Auteur: Sander Hoogendoorn

Back to the industry.
I'll admit. Sometimes, when I'm bored, I browse through LinkedIn, looking for something worthwhile to read or add vicious comments to. The most annoying category of posts on LinkedIn, next to advertisements and silly AI-generated posts, must be the endless stream of people posting about certificates they have just been awarded. Some have become professional Scrum Master (PSM) or Professional Scrum Master II (or even III). Others have achieved certificates for Java, C#, testing, DevOps, project management, SAFe, or even mastering Jira. You name it, and it exists.

There is no end to the certificates you can receive.
Don't get me wrong. There's a place and time for certifications in our industry. Certificates can provide a structured path for learning, and the training towards them can help professionals gain foundational knowledge in specific areas. Sure. However, I often see certificates that can be achieved too easily. For example, most people have little trouble reading the Scrum guide, which is about fifteen pages, and consequently pass a Scrum Master certification exam. Last time I checked, you needed to fill in a questionnaire with thirty multiple-choice questions. That's it.

Without having any experience whatsoever.
Abd that's what I have issues with. Running training courses and handing out certifications has become a big industry, especially in tech. There's simply too much money to be made for certifying "institutes." And when everybody has passed their exams, and the money dries out, they just invent new ones.

You start off with three-letter acronyms (PSM) and then move to

BIO

Sander Hoogendoorn

Sander is an independent dad and traveler. A seasoned developer with over four decades of experience and still daily writing code, Sander has survived in the tech world in various roles, from CTO of companies like iBOOD, ANVA, and Klaverblad to being Capgemini's global agile thought leader.

Known for his post-agile mindset and provocative perspectives, Sander empowers organizations and teams to break free from the norm and embrace innovation. He's not just about writing code; he's about rewriting the rules.

four-letter acronyms (PSPO). Or add Latin numbers (such as in PSM II). The worst I think I saw was a few weeks ago when someone actually posted that they had earned the Professional Scrum Product Backlog Management Skills badge. I guess there's always more money to make.

And all this can still be achieved without having any tacit experience.
It's not just Scrum. A friend, who is extremely good at learning, recently decided to pick up on his certificates for a well-known global internet infrastructure provider, as someone in their organization needed to be certified. He read the documentation thoroughly and passed several exams, leading to six certificates. He is now an Edge DNS & Basic App Enablement Partner Support Engineer who has never configured anything in infra his whole career.

Or, imagine you're coming from outside the tech industry and looking to land a good job. What better way than to get your Professional Scrum Master certification and, based on that, be hired to coach a team of ten senior developers? How does that go, you think? Too often, I've witnessed this situation. Literally.

Based on certificates, managers hire people who hardly understand who and what they're coaching. I've had coaches coming to me in disbelief, saying "that the team doesn't want to be coached anymore." Or developers dragging themselves to retrospective meetings just because they think they have to. It's sad. Both for the coaches and for the developers they're coaching.

Certificates should complement, not replace, hands-on experience. Too often, I've seen people armed with arrays of certifications flounder when confronted with real-world problems because they lack the practical skills to back up their credentials.

Please, never underestimate the value of experience. Your time and effort in dealing with challenging projects, troubleshooting issues, and collaborating with others is invaluable. It strengthens your problem-solving skills, enhances your adaptability, and creates a deeper understanding of the tech and the people you work with. When I got my swimming certificates, at least I had a fair amount of practice. Imagine if I wouldn't have.

Experience beats certification. Every single time.
One last tip. If you find yourself compelled to pursue certification just because managers or recruiters are demanding, create your own. With a rapidly growing number of organizations figuring out that there is money in certification, an endless stream of non-sensical certifications is flooding the industry. Crafting your own certificate may not be as outlandish as it sounds. I know I did. I'm a Certified Flow Resource (CFR). Look up the video of my talks about this topic.

Just make sure you come up with a catchy three or four-letter acronym to rival the likes of the industry giants. But we've always been good at naming in our industry, so that shouldn't be a problem, right? ●

Functionaliteit gefaseerd live zetten met feature flags

Als ontwikkelaars voegen we regelmatig functionaliteit toe aan bestaande applicaties, vaak is het hierbij lastig in te schatten hoe gebruikers erop gaan reageren. Met feature flags kunnen we gefaseerd functionaliteit uitrollen om feedback te verzamelen.

Auteur: Maarten Grootoonk

Via feature flags kan je met beperkte impact toetsen hoe wijzigingen in productie presteren, ook kun je een beeld krijgen van wat gebruikers van nieuwe ontwikkelingen vinden. Met feature flags kunnen wijzigingen en functionaliteiten namelijk gefaseerd uitgerold worden naar een deel van de gebruikers of voor gebruikers met specifieke eigenschappen.

Met de inzichten van een gedeeltelijke uitrol kan een onderbouwde keuze worden gemaakt om een wijziging wel of juist niet uit te rollen naar de hele gebruikersgroep.

Met de App Configuration dienst van Azure kun je feature flags aanmaken en in real-time configureren. Via de Azure App Configuration SDK kunnen de waardes van deze feature flags worden uitgelezen. Er zijn SDK's beschikbaar voor verschillende populaire programmeertalen.

Feature flag aanmaken
Een feature flag is in de basis een boolean die uit of aanstaat. Deze boolean geeft bijvoorbeeld aan of een nieuwe pagina zichtbaar moet worden of dat een nieuw algoritme actief wordt.

We starten voor het maken van de eerste feature flag in Azure, hier maken we een Azure App Configuration resource aan. Na het aanmaken van de resource kunnen we de eerste feature flag aanmaken. Bij een feature flag is de naam en aan/uit boolean verplicht. Optioneel kunnen er labels en/of een feature filters worden toegevoegd. **Figuur 1**

Na het aanmaken van de Azure resource kan er vanuit .NET code verbonden worden met de Azure app Configuration resource om de feature flags op te halen.

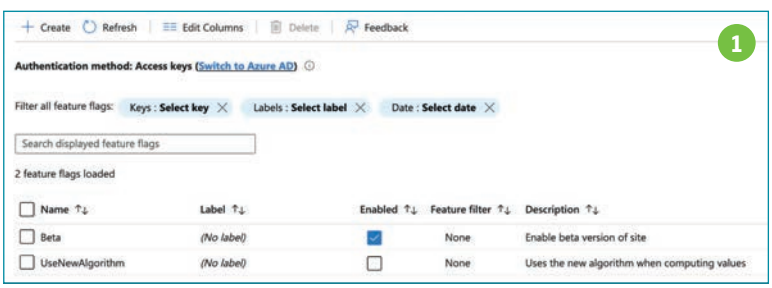
De eerste stap is het installeren van de volgende twee NuGet packages, deze bieden extension methods aan om Azure App Configuration in te stellen. **Figuur 2**

Een van deze methoden is de AddAzureAppConfiguration methode, hiermee kan onder andere de connectie parameter worden ingesteld. **Figuur 3**

Feature flags in code

Nu we Azure App Configuration hebben geconfigureerd, kan in de applicatie logica opgevraagd worden of een bepaalde feature flag uit of aanstaan staat.

Feature flag waardes kunnen worden uitgelezen door de IFeatureMana-



Azure App Configuration feature flag overzicht.

Azure App Configuration NuGet packages.

```
builder.Configuration.AddAzureAppConfiguration(options =>
{
    options
        .Connect(builder.Configuration.GetConnectionString("AppConfig"))
        .UseFeatureFlags();
});
```

Verbinding maken met Azure App Configuration.

```
[ApiController]
[Route("[controller]")]
public class ThemingController : ControllerBase
{
    private readonly IFeatureManager _featureManager;

    public ThemingController(IFeatureManager featureManager)
    {
        _featureManager = featureManager;
    }

    [HttpGet]
    public async Task<string> Index()
    {
        if (await _featureManager.IsEnabledAsync("Beta"))
        {
            return "Beta version of site";
        }

        return "Regular site version";
    }
}
```

Feature flag waarde uitlezen met IFeatureManager.

```
<h1>Hello world!</h1>

<feature name="Beta">
    <p>Beta feature is live!</p>
</feature>

<p>This is page content</p>
```

Feature flag check in razor view.

```
app.UseAzureAppConfiguration();
```

Azure App Configuration middleware activeren.

ger service via dependency injection te injecteren en de IsEnabledAsync methode aan te roepen. **Figuur 4**

Naast het aan- en uitzetten van delen van applicatie logica kunnen ook stukken van razor views worden getoond op basis van de feature flag waarde. In Figuur 5 wordt het "Beta feature is live!" stuk alleen getoond als de "Beta" feature flag actief is. **Figuur 5**

Feature flags wijzigingen

Standaard worden de waardes van de feature flags ingeladen bij het opstarten van de applicatie. Wijzigingen aan de feature flags hebben hierdoor pas effect als de applicatie wordt herstart.

Vaak willen we dat wijzigingen aan feature flags meteen actief zijn omdat we dan sneller functionaliteit uit/aan kunnen zetten. Ook hoeven we dan de applicatie niet te herstarten.

Het automatisch herladen is mogelijk met de Azure App Configuration middleware, deze middleware is te gebruiken via UseAzureAppConfiguration methode. Via deze middleware wordt bij een http-aanvraag gekeken of er een nieuwe waardes zijn voor de feature flags. Deze opgehaalde waarden wordt standaard gecached voor een in te stellen tijd. Dit om te voorkomen dat er bij elke http aanvraag een aanvraag naar Azure gaat. **Figuur 6**

Als je een korte cache tijd instelt zijn wijzigingen aan een feature flag waarde sneller zichtbaar bij gebruikers. Echter zorgt dit wel voor meer aanvragen naar Azure wat kosten met zich mee kan brengen. **Figuur 7**

Dynamische Feature flags

In veel gevallen wil je applicatie wijzigingen gefaseerd uitrollen, idealiter krijgen eerst een aantal gebruikers de update en kan op basis van de feedback worden bepaald of de wijziging naar iedereen uitgerold kan worden.

Het aan/uit zetten van feature flags voor een percentage van gebruikers en/of op basis van eigenschappen van de ingelogde gebruiker kan via Feature Filters.

Om Feature Filters te kunnen gebruiken moet dit worden aangezet met de AddFeatureFilter extension methode. Bij het gebruik van een TargetingFilter moet ook een implementatie van de ITargetingContextAccesor worden geleverd. **Figuur 8**

De ITargetingContextAccesor implementatie schrijf je zelf, hierin geef je Azure App Configuration informatie over de ingelogde gebruiker. Zo kun je aangeven bij welke groepen de gebruiker hoort en wat zijn/haar unieke identifier is. Deze groep na-

```
builder.Configuration.AddAzureAppConfiguration(options =>
{
    options.Connect(builder.Configuration.GetConnectionString("AppConfig"))
        .Select("TestApp:*", LabelFilter.Null)
        // Configure to reload configuration if the registered sentinel key is modified
        .ConfigureRefresh(refreshOptions =>
            refreshOptions.Register("TestApp:Settings:Sentinel", refreshAll: true));
});
```

Cache periode instellen.

```
builder.Services.AddFeatureManagement()
    .AddFeatureFilter<TargetingFilter>();

builder.Services.AddSingleton<ITargetingContextAccessor, CustomTargetingContextAccessor>();
```

Feature flags configureren voor dynamische feature flags.

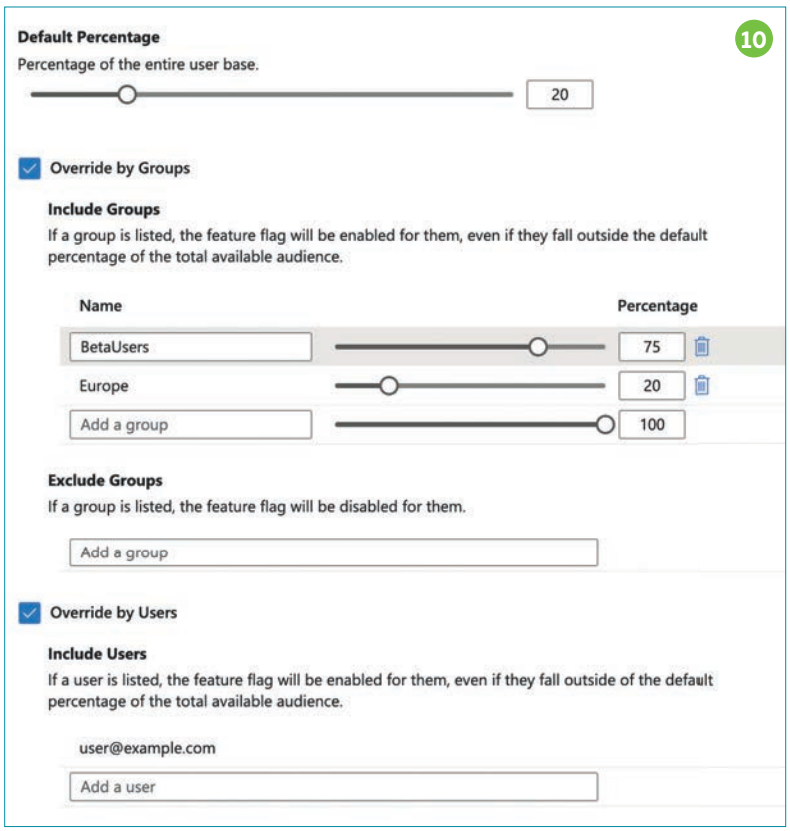
men kun je zelf bedenken, dit kunnen rollen maar ook zelf samengestelde groepen zijn.

In Figuur 9 geven we via de TargetingContext informatie aan de Azure App Configuration middleware, zo plaatsen we de gebruiker in de BetaUsers groep op basis van onze eigen logica. Deze logica kan bijvoorbeeld in de database kijken of de gebruiker een Beta optie binnen de applicatie heeft aangevinkt. **Figuur 9**

Feature flags kunnen bij vrijwel elke applicatie helpen met het gefaseerd uitrollen van features om vervelende verrassingen te minimaliseren.

Via TargetingContext informatie vastleggen over de huidige gebruiker.

In de Azure omgeving kan dan precies worden ingesteld voor welke gebruikers en groepen de feature flag moet gelden. Deze configuratie wordt in Azure gedaan zodat je op elk moment de configuratie kan wijzigen, zo kun je de feature bijvoorbeeld naar 100% van de gebruikers uitrollen als de feedback positief is.



Via Azure een Feature Filter configureren met verschillende parameters.

Hierdoor zijn er geen code wijzigingen of nieuwe deployments nodig. **Figuur 10**

Zelf aan de gang

Feature flags kunnen bij vrijwel elke applicatie helpen met het gefaseerd uitrollen van features om vervelende verrassingen te minimaliseren. Op de Microsoft Learn documentatie site zijn er verschillende tutorials om Azure App Configuration te integreren in applicaties. ●

BIO

Maarten Grootoonk

Maarten Grootoonk is een Software Engineer en Trainer bij Ordina die zich specialiseert in het Microsoft landschap. Zo ontwikkelt hij .NET applicaties voor de klant en geeft hij verschillende Azure trainingen en workshops.


THE WORLD NEEDS MORE INNOVATORS
A team of six people created Tikkie. After successfully launching an app like Tikkie, it would be easy to say "Okay, we made it". But that's not how innovation works. Innovation is about constantly challenging yourself and spotting opportunities. Daring to acknowledge that something can always be better. Because it's this mindset that makes the difference. And to make a difference we need more innovators – people like those who developed Tikkie and Groepie. We need you. Employees that will continue to motivate others, and us.
www.workingatabnamro.com
ABN·AMRO

Moet ik naar **.NET 8 upgraden?**

In het kort / TL;DR

Ja, upgraden is zeer aan te raden. Alleen een product dat binnenkort end-of-life is, zou ik niet meer upgraden. Een upgrade naar .NET 8 zal je gebruikerservaring ten goede komen, geeft je op allerlei vlakken een concurrentievoordeel en zal je ontwikkelteam ook nog eens sterk motiveren. .NET 8 kan je dit bieden, omdat het de wendbaarheid zal verbeteren, je performance op ontzettend veel vlakken zal verbeteren en opent daarbij een aantal nieuwe functionaliteiten die je kunt benutten ten goede van je product en/of dienst. Daarbij: je ontkomt er simpelweg niet aan om te upgraden. Dan kun je het maar beter snel achter de rug hebben en profiteren van de voordelen.

Auteur: Rene Elstgeest

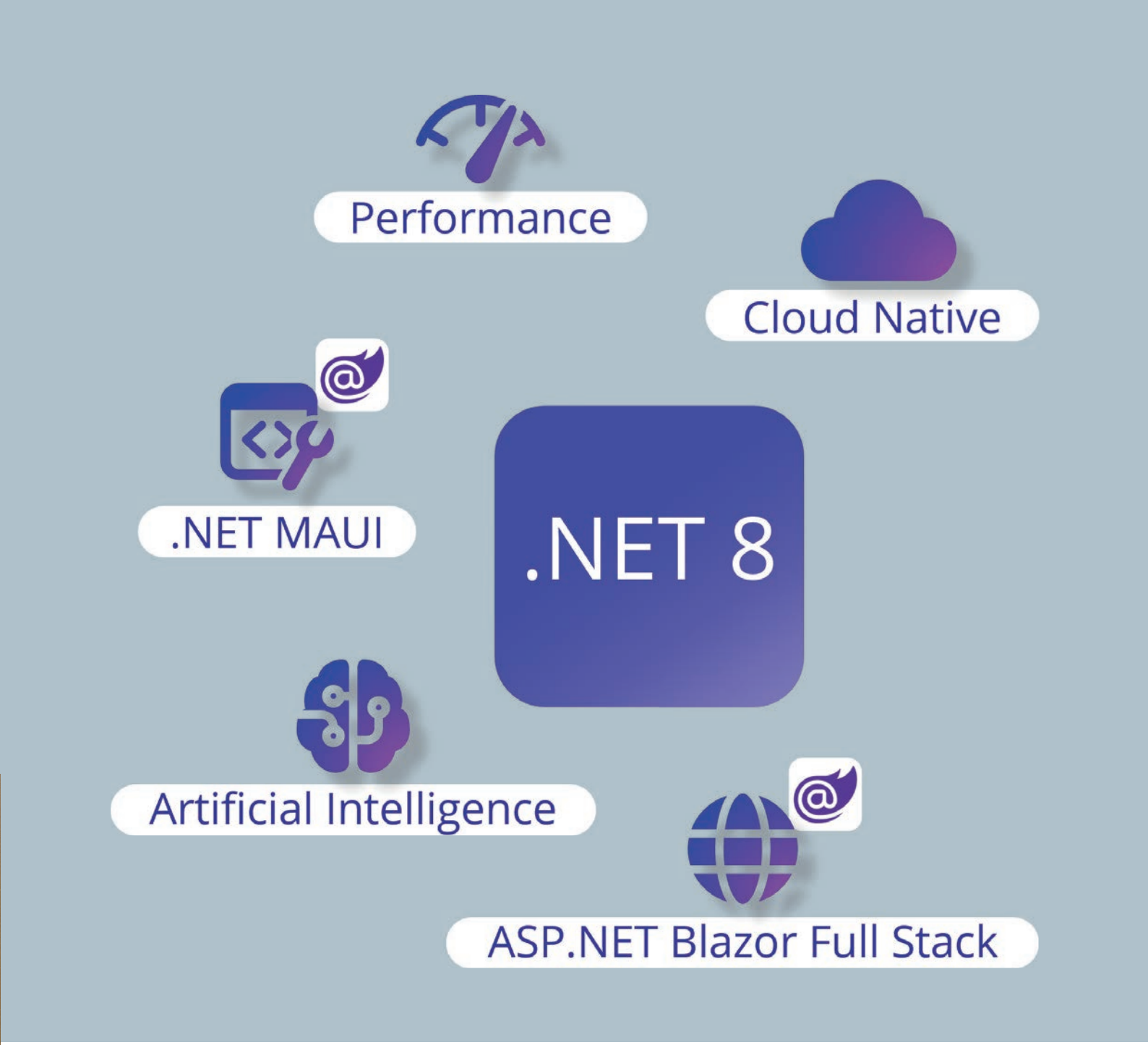
Concurrentievoordeel

Het belangrijkste overkoepelende voordeel van de nieuwste stabiele versie van .NET te gebruiken in je software is dat je voordeel ten opzichte van je concurrent zo snel mogelijk meepakt. Hoe eerder je dat doet hoe eerder je voordeel hebt van een betere gebruikerservaring, beveiliging en de professionele tevredenheid. Die upgrade zal vaak toch wel een keer moeten plaatsvinden, waarom dan niet zo vroeg mogelijk?

Verbeter je gebruikerservaring
Niemand houdt ervan om te wachten. Naast dat het tijd kost om te wachten, verstoort het je werkritme en raak je moeilijk, of helemaal niet, in je eigen "flow". Taken die je uitvoert wil je vaak graag zo snel mogelijk doen met voldoende aandacht voor een correcte uitvoering ervan. Onnodig wachten wekt irritatie op wat ten koste gaat van zowel de snelheid als de scherpte waarmee mensen hun werk kunnen uitvoeren. In het geval van webshop of entertainment applicaties is de gebruikerservaring nog veel belangrijker. Is de interactie ook maar een beetje te langzaam? Bezoeker weg. .NET 8, alsmede zijn recente voorgangers, verbetert opnieuw zijn performance die je voor een groot gedeelte nagenoeg gratis krijgt. Je hoeft me niet op mijn woord te geloven hier en hier voor ASP.NET kun je de technische details nalezen, inclusief onderbouwende benchmarks. Het is echt aan te raden dit te (laten) doen indien je nog twijfelt.

Beveiliging

Vanuit beveiligingsoogpunt is het altijd belangrijk om de meest recente versie te hebben draaien. De laatste security patches zijn wel echt heel essentieel, maar ook de major versies bieden op gebied van beveiliging ook steeds betere features. Het voordeel van de laatste major versie wel up-to-date houden is dat je dan nul risico loopt afhankelijk te worden van support die er niet is of waar je zelfs extra voor moet betalen. Je kunt simpelweg niet zonder deze support vanuit beveiligingsoogpunt. Organisaties die .NET Framework bijvoorbeeld nog draaien lopen serieus gevaar te verzanden in een tekort aan partijen die hier nog professionele ondersteuning kunnen bieden. Dat is niet een ideale situatie. Vernieuwing van je software pakket wordt op deze manier wel erg



lastig om uit te voeren. Nieuwe features die worden uitgebracht en worden verbeterd maken het daarbij regelmatig eenvoudiger voor de ontwikkelaar maken om applicaties goed te beveiligen.

Motivatie van het ontwikkelteam

Een andere belangrijke reden om met de laatste software te werken is dat het goede ontwikkelaars aantrekt. Niemand houdt ervan het wiel te ontwikkelen dat al bestaat. Helaas gebeurt dat wel een heel stuk sneller als je niet de laatste features, mogelijkheden en tools bij de hand hebt. Het eenvoudiger maken om programmatuur duurzaam te kunnen structureren zijn vaak belangrijke voordelen van de nieuwste versies. Nieuwe features van programmeertalen, zoals van C# 12, geven ontwikkelaars nieuwe mogelijkheden om efficiënter, effectiever en duurzamer code te schrijven. Oplossingen als Blazor kunnen bijvoorbeeld het voordeel bieden dat je slecht één programmeertaal nodig hebt om een web applicatie te schrijven, namelijk C# (en natuurlijk voor de opmaak nog een beetje HTML en (S)CSS). Kortom, met de nieuwste versie van .NET geef je je ontwikkelteam zowel een motivatie als een efficiëntie boost, zonde om niet mee te pakken. ●

Twee benaderingen **voor het vergroten van impact**

Je wilt aan de slag om het team te verbeteren, maar waar begin je eigenlijk? Dan is het goed om te realiseren hoe veranderingen werken. Dit artikel beschrijft 2 methodieken die je kan toepassen om de gezondheid en het presteren van teams te verbeteren.

Auteur: Arjen Kraak & Martijn Broos

De methodiek van '6 types of Working Genius' belicht het proces waarover werk wordt doorlopen en welke competenties hierbij nodig zijn. Dit is de persoonsgerichte benadering.
De methodiek van '5 frustraties van werken in teams' belicht de evolutie naar het optimaal samenwerken met een teamdoelstelling en beloftes. Dit is de teamgerichte benadering.

De persoonsgerichte benadering: '6 types of working genius'

Elke activiteit (zakelijk of privé), hoe klein ook, wordt doorlopen in een aantal stappen. Het uitvoeringsproces kan worden weergegeven als onderstaande metafoor.

- WIDGET staat voor het in elkaar grijpen van stappen in het uitvoeringsproces van elke activiteit. Deze stappen zijn:
- Wonder (verwondering, afvragen); dit is de stap waarin men zich afvraagt "waarom werkt het zo" of "kunnen we hier iets op verzinnen". Dus de stap waarbij een soort van kans wordt geroepen voor een activiteit/verbetering. De persona die bij Wonder hoort is de dromer. De belangrijkste vraag die de dromer zich stelt is "waarom?".
- Invention (innovatie, ontwerp); dit is de stap waarbij men spontaan met oplossingen komt voor de gestelde kans. Dit zijn vaak de momenten van "oh, dat kunnen we zo oplossen, simpel toch?". De persona die bij Invention hoort is de uitvinder. Een uitvinder denkt in nieuwe manieren om verbeteringen te realiseren; Hierbij is hij vaak optimistisch en denkt in kansen.
- Discernment (verfijnen, aanscherpen); dit is de stap waar men zich de gevolgen van de oplossing gaat afvragen of aan alles is gedacht. De persona die bij Discernment hoort is de realist. Waar een uitvinder denkt in kansen en mogelijkheden, is de realist juist kritischer. Vaak denkt de realist in "maar" terwijl hij moet denken in "en".
- Galvanizing (motivatie, verzamelen); dit is de stap waar men de manschappen verzamelt om de activiteit uit te dragen. Betrokkenen te motiveren en achter de kans en de oplossing gaan staan. De persona bij galvanizing is de motivator. Het is een teamspeler die kan overtuigen en enthousiasmeren.
- Enablement (faciliteren, adviseren, coachen); dit is de stap tijdens de uitvoering van de activiteit waarbij men in staat wordt gesteld het werk te kunnen doen. Zorgen dat diegenen die de activiteit tot uitvoer brengen hun werk kunnen doen. De persona die bij enablement hoort is de helper. Hij stelt zich in dienst van elk teamlid en regelt vooral dat aan randvoorwaarden wordt voldaan en verwijdert belemmeringen.
- Tenacity (doorzetten, taken "aftikken"); dit is de stap waarin daadkrachtig actiepunten met focus en prioriteit worden afgerond. De persona bij tenacity is de uitvoerder. De uitvoerder heeft een sterke wil om (taken)lijstjes af te vinken. Uitvoerders zijn minutieus en gedetailleerd.

Het blijkt dat elk individu zijn of haar talenten het beste benut voor twee stappen in het WIDGET uitvoeringsproces. Ook is het zo, dat elke persoon talenten ontbeert voor twee andere stappen in het WIDGET uitvoeringsproces. Bij langdurig uitvoeren van werkzaamheden in deze stappen bestaat de kans op burn-out klachten. Tijdens het Bergler webinar 'Professionals vinden en behouden' vanaf 28:10, zie je de gevolgen voor je team en welke persona's welke invloed kunnen hebben.

Elke radertje in het proces is even belangrijk. Dit betekent ook dat talenten

van mensen even belangrijk zijn om tot een bepaald resultaat te komen. Veel belangrijker is dat elk talent aanwezig en beschikbaar is binnen een team.

De teamgerichte benadering: “De 5 frustraties van samenwerken”

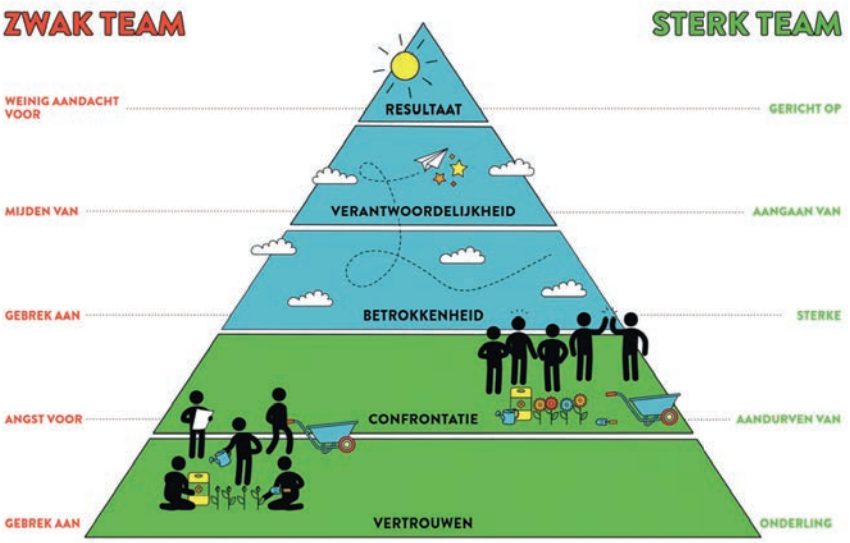
De ‘5 frustraties’ van een team is een raamwerk dat helpt om barrières naar effectief werken in een team weg te elimineren. Het ultieme niveau van effectief werken is het bereiken van gezamenlijk werken aan team doelstellingen.

Volgens het model zijn de 5 frustraties gestapeld als een piramide. Het idee is dat het onderliggende niveau moet zijn opgelost alvorens het volgende niveau kan worden bereikt. Het raamwerk biedt oefeningen om inzicht en verbeteringen te identificeren om tot het volgende niveau te komen.

> Afwezigheid van vertrouwen; elke menselijke relatie begint met vertrouwen. Dat betekent niet alleen dat je op elkaar kunt bouwen in goede en slechte tijden. Ook, en misschien belangrijker, betekent het dat je in veiligheid onderwerpen kan delen zonder daarop persoonlijk of in het team voor wordt afgerekend.
> Angst voor conflicten; het vertrouwen en de veiligheid is er. Echter bestaat er angst deze basis te verliezen bij het uitspreken van verschillen van mening. Een team moet op constructieve wijze conflicten hebben over de inhoud als fundament voor hogere lagen in de piramide.
> Afwezigheid van commitment; in deze laag werken teamleden nog op individuele doelstellingen. Er is onvoldoende bevlogenheid om de doelstellingen van het team te realiseren. Eigen doelstellingen gaan voor team doelstellingen.
> Ontwijken van verantwoordelijkheid; het team ontwijkt elkaar bij het aanspreken op de verantwoordelijkheden die iedereen heeft om de doelstellingen te bereiken.

De twee benaderingen, is het ‘of’ of ‘en’

Zoals eerder gesteld zijn de benaderingen gericht op de persoon en het team. Aangezien teams bestaan uit personen kunnen de benaderingen



Arjen Kraak
Interim CTO,
Bergler

Martijn Broos
Tech lead,
Bergler

niet los van elkaar worden gezien. Om impact te verhogen begin je bij de persoon, dus de ‘6 types of working genius’. Elk lid van het team heeft twee talenten waarin men uitblinkt. Daarnaast zijn er twee talenten die ervoor zorgen dat bij langdurige blootstelling en uitvoering een teamlid afbrandt. De workflow van werk door het team wordt vooral beïnvloed door de balans van de persoonlijke talenten in het team. Als een team bestaat uit alleen ‘dromers’ en ‘uitvinders’ zal het team onvoldoende producten opleveren. Als een team bestaat uit alleen ‘uitvoerders’ zal er onvoldoende worden nagedacht over verbeteringen en innovatie. Deze benadering respecteert het talent van mensen, er bestaat geen werkvorm om mensen te transformeren naar een ander talent.

Zodra er balans van talenten aanwezig is, kan het team werken aan de frustraties van samenwerking. Het verhogen van impact wordt dan vervolgd met de methodiek van ‘5 frustraties van samenwerken’. Om tot optimale impact te komen moeten alle lagen van de frustraties worden geëlimineerd. Voor deze benadering zijn werkvormen beschikbaar die aansluiten bij het boek. Zie hiervoor de bronvermelding.

Samengevat, het is dus EN. Het verhogen van impact begint met de persoonlijke benadering die als fundament dient voor de samenwerking benadering. ●

Bronvermelding:
6 Types of Working Genius door Patrick Lencioni (boek en assessment) (EAN: 9781637743294)
5 Dysfunctions of a team door Patrick Lencioni (boek en werkboek) (EAN: 9780787960759)

Jasper Sprengers

De rampzalige illusie van perfecte software

Met ergernis door slechte software worden we dagelijks geconfronteerd, maar het debacle bij het Britse Post Office is er een van de buitencategorie. Vanaf 1999 werden 900 filiaalhouders vervolgd voor grootschalige fraude en belandden honderden achter de tralies. In bijna alle gevallen waren bugs in het boekhoudpakket Horizon de schuldige achter het vermeende gesjoemel. De leiding was op de hoogte en verdient daarmee de grootste blaam voor de gerechtelijke dwaling. Met miljarden aan compensatie mag de regering het beschaamde vertrouwen in de rechtsstaat nu herstellen, voor zover dat nog kan.
Meteen vallen je de overeenkomsten op met onze eigen toeslagenaffaire. Arrogantie van de macht die uitdraait op faillissementen, echtscheidingen, tot zelfdoding aan toe. Groot verschil was wel dat onze politiek een draconische wet had bedacht en de rechterlijke macht nul coulance toonde in de uitvoering. Het was in essentie geen softwareprobleem. De opsporingsalgoritmes met hun discriminerende voorkeur maakten het weliswaar erger, maar werkten verder as designed. Horizon van leverancier Fujitsu was daarentegen zo lek was als het spreekwoordelijke mandje. Als je vervolgens niet mocht twijfelen aan de kwaliteit krijg je cognitieve dissonantie. Zie maar: kort nadat je landelijk een nieuw boekhoudsysteem uitgerold hebt kloppen de cijfers van honderden kantoren niet meer. Hoe is dat ineens mogelijk? Brengt een boze geest plotseling al die hardwerkende franchisenemers massaal op het slechte pad? Natuurlijk had de Post Office niet zo'n naïef en verknipt mensbeeld. Alles wees op fouten in de software. Maar in een laffe poging hun eigen straatje schoon te vegen moesten de onschuldige postmasters hangen. Dit waren mensen die iedereen kende en waardeerde in de gemeenschappen waar ze hun nering hadden. Toch waren bugs hier enkel de lont in het kruitvat. Als de rechtsstaat zijn beloop had gehad was het tenminste bij een financiële ramp gebleven. De Post Office had zich de illusie van perfecte software laten verkopen. Ik ken de cultuur van het Japanse Fujitsu niet, maar ik denk niet dat men er ruimhartig en constructief omging met fouten. Nu draait de doorsnee lezer van dit blad niet aan de knoppen van de democratie, maar als makers van software kunnen wij wel degelijk bijdragen aan een cultuur die dit soort rampen helpt voorkomen. Weg dus met de mythe dat er überhaupt zoiets bestaat als onfeilbaar technisch vernuft binnen een team of individu. Stoot de arrogante nerd van zijn voetstuk – en ja, het zijn altijd mannen. Geef toe dat jij met je twintig jaar ervaring nog steeds uitglijers maakt. Foutloze software schrijven is gewoon hartstikke lastig. Er is eigenlijk maar één verhaal dat we in alle eerlijkheid aan de klant mogen vertellen. "We hebben ons uiterste best gedaan om kwaliteit te leveren, maar het kan altijd dat we steken hebben laten vallen. Dus laat ons alsjeblieft meteen weten als je iets geks ziet, dan gaan we het oplossen". Maar ja, in de macho hackercultuur waar niemand durft toe te geven dat iets hun boven de pet gaat klink je dan meteen weer als zo'n watje. ●

Jasper Sprengers

“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
a Sopra Steria company

- 1 jaar intensieve training voor developers, door Microsoft Certified Trainers (MCT) en Microsoft Most Valuable Professionals (MVP)
- Uitsluitend moderne technieken
- Combinatie van vakinhoud en soft skills

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

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

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

Microsoft • .NET • AZURE • DevOps • C# • Cloud • AI/ML • Security • PowerPlatform • Copilot