SDN MAGAZINE

Nummer
146
Mrt 2023

SDN Magazine is
een uitgave van:
SDN

Van en voor Microsoft & .NET professionals

- Future Tech Special
- Test Containers
- Intervisie in de praktijk

SELMA HELPS 1.3 MILLION PEOPLE MAKE ENDS MEET

How many people do you want to help?

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

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

Take a look at our open roles:

netcompany

Colofon

Uitgave
Software Development Network
27ste jaargang
Nr. 146 • maart 2023

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

Auteurs:
Florian Van Dillen, Martijn Broos, Raymond Jetten, Erik van Olst, Arjen Kraak, Jacob Duijzer, Stefan van Tilborg

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

Beste SDN-ers,

Met het SDN-magazine hebben we het tempo weer te pakken. In 2022 is het ons gelukt om je elk kwartaal te voorzien van een mooi opgemaakt en goed gevuld magazine. Fijn dat nieuwe auteurs ons mooie magazine voorzien van interessante artikelen uit de .NET community. Wij zijn er erg blij mee, laat je feedback horen via @SDN_Community of via redactie@sdn.nl. Mocht je wensen hebben voor een onderwerp of een startende/ervaren schrijver kennen, breng hem/haar dan ook met ons in contact.

Ook met Futuretech (https://futuretech.nl/) hebben we het tempo. Op 15 maart is ons volgende Futuretech event! En als SDN-abonnee ga je gratis naar dit event, naast een gaaf programma meer dan voldoende ruimte om te netwerken met je peers uit de community. Verderop in dit magazine vind je meer informatie over het programma. Het wordt heel gaaf met de eerste editie van de SDN-Pubquiz, kom langs en deel je kennis.

Voor 2023 hebben we sowieso grootste plannen, naast vier magazines willen we ook met een paar events komen. We zijn ook nog bezig met iets heel anders, de Masters of .NET. Dat wordt echt leuk, dan kun jij met een team laten zien hoe jullie .NET tot het uiterste inzetten. Meer informatie volgt zodra we het verder hebben uitgewerkt, medio 2024 wordt dit het Microsoft .NET event van het jaar!

Zoals je ziet bruist de SDN als nooit tevoren! Reden genoeg om abonnee te worden van deze geweldige community. En je krijgt er veel voor terug, magazines en gratis toegang tot Futuretech.

In dit magazine hebben Florian, Martijn, Jacob, Arjen, Erik, Raymond en Stefan hun kennis voor jullie in een artikel gegoten over Contianers, C4, Intervisie, DAPR en Cybersecurity alles om jou Future proof te houden! Geniet en laat je inspireren.

Veel leesplezier en komt langs bij de SDN-cast op Futuretech!

Tot de volgende editie en bij Futuretech,

Marcel

BERGLER

SOFTWARE SOLUTIONS

KENNIS IS MACHT, KENNIS DELEN IS KRACHT!

BERGLER SOFTWARE SOLUTIONS IS OP ZOEK NAAR ERVAREN

LEAD .NET DEVELOPERS.

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

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

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

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

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

www.bergler.nl

May the source be with you!

In dit nummer

6 Wanneer je integratietesten bouwt zul je vroeg of laat moeten beslissen op welke manier je de afhankelijkheden van je tests gaat aanspreken. Dit kan een database zijn zoals MySql of MsSql, maar ook een Redis cache of een RabbitMQ message broker. Naast de gebruikelijke oplossingen is er ook een andere manier, gebaseerd op Docker containers.

15 FUTURE TECH 2023
THE IT CONFERENCE FOR MICROSOFT TECHNOLOGIES
Special

25 Dapr ofwel Distributed Application Runtime biedt een framework waarbij de complexiteit van Service Discovery, Service Bus integratie, secrets management, observability en encryptie uit handen genomen wordt, zodat de developer zich kan focussen op het schrijven van business logica.

Test Containers

Wanneer je integratietesten bouwt zul je vroeg of laat moeten beslissen op welke manier je de afhankelijkheden van je tests gaat aanspreken. Dit kan een database zijn zoals MySql of MsSql, maar ook een Redis cache of een RabbitMQ message broker. Naast de gebruikelijke oplossingen is er ook een andere manier, gebaseerd op Docker containers.

Auteur: Florian Van Dillen

Testcontainers

Testcontainers zijn kleine wegwerp-containers die je kunt gebruiken voor je integratietests. Zodra de tests worden gedraaid wordt er bij het opstarten een container klaargezet die je kunt gebruiken om bijvoorbeeld een database of cache in te draaien. Je tests kunnen hier dan gebruik van maken. Zodra de tests klaar zijn worden de containers automatisch weggegooid. Er blijft dus niets bewaard van de containers.
Deze functionaliteit is eenvoudig in te bouwen in ieder .NET project (**voorbeeld 1**). Je kunt hiervoor gebruik maken van de testcontainers-dotnet library. Dit is een library die ook voor veel andere talen beschikbaar is, zoals Java, Go en Node.js.
Het leuke van de library is dat je puur C# kunt schrijven. Er komen geen Docker API's aan te pas. Alleen als je dat echt wil zou je met een Dockerfile kunnen werken. Meer hierover onder het kopje "zelf een image maken".

Waarom testcontainers

Er zijn verschillende mogelijkheden om integratietests te draaien. Iedere manier heeft zijn eigen voor- en nadelen.
Zo kun je gebruik maken van een gedeelde database. Dit is vaak een database die op een VM of in Azure draait, waar alle developers binnen een team gebruik van kunnen maken. Het voordeel hiervan is dat deze database er altijd is, je hoeft dus geen setup te doen vanuit je code anders dan het instellen van een connection string.
Het nadeel van deze aanpak is dat je na het verbinden eerst de database in een staat moet brengen waarin deze klaar is voor de tests. Dit kunnen schema upgrades zijn, maar ook het resetten van alle data die nodig is voor de tests.
Ook is het nodig om na afloop de database weer in oorspronkelijke staat terug te brengen, zodat je andere teamleden of zelfs verschillende integratietests in je project niet in de weg gaat zitten.

Een andere mogelijkheid is het opbouwen van ad-hoc infrastructuur, bijvoorbeeld met ARM of Bicep in Azure. Het voordeel is dat je een schone database hebt bij iedere testrun, waardoor tests elkaar niet in de weg zullen zitten.
Het nadeel is dat je moet wachten op de infrastructuur en dat je hiervoor

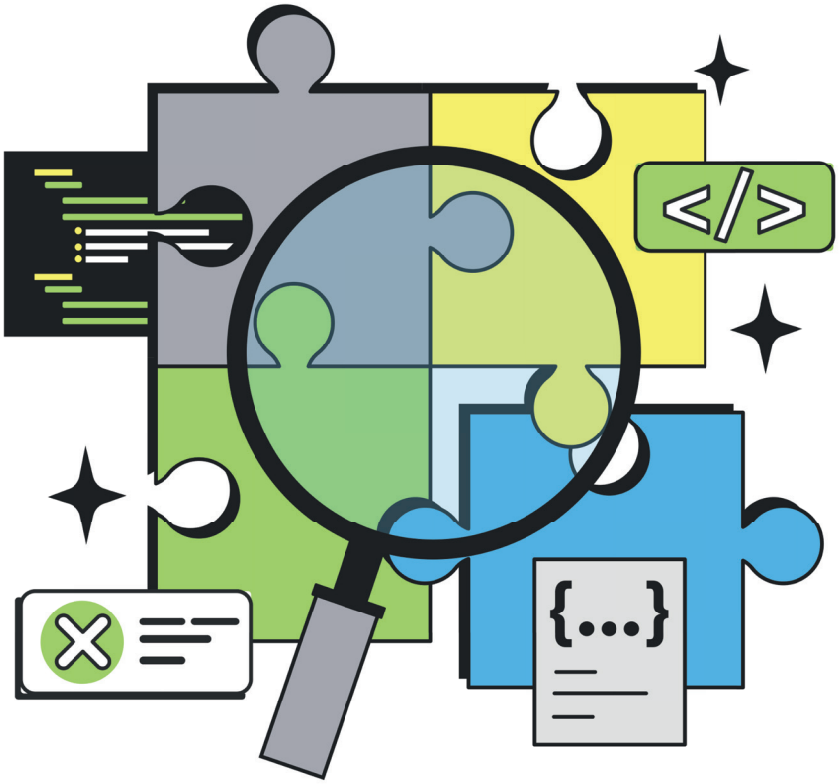
```
dotnet add package Testcontainers --version 2.2.0
```

1

BIO

Florian van Dillen

Florian van Dillen is een full-stack developer bij 4Dotnet. Hij werkt het liefste met de nieuwste cloud-technieken zoals containers, serverless en messaging componenten. Daarnaast heeft hij interesse in architectuur en informatiebeveiliging. Momenteel is hij voor klanten bezig met IoT en Azure Container Apps.



extra code moet schrijven. Ook moet je na afloop de infrastructuur weer weggooien, anders blijft de meter in Azure lopen.

Testcontainers combineert de voordelen van de twee eerdere oplossingen: je hebt altijd een schone database, indien nodig meerdere databases of zelfs per test method een eigen database. Je bent relatief kort aan het wachten op het opstarten van de container en na afloop wordt deze volledig automatisch weer weggegooid.

TestcontainersBuilder

De Testcontainers library doet al het zware werk voor je. De Fluent API van de TestcontainersBuilder zorgt ervoor dat de meeste containers met slechts enkele regels code op te bouwen zijn (**voorbeeld 2**). Voor veel voorkomende afhankelijkheden zijn kant-en-klare modules beschikbaar. Daardoor is er nog minder configuratie nodig, wat tijd scheelt bij het bouwen van de tests.
Een volledige lijst met kant-en-klare modules is te vinden op: https://dotnet.testcontainers.org/modules/ ●

Testcontainers combineren schone databases en korte wachttijd

2

```
var container = new TestcontainersBuilder<MsSqlTestcontainer>()
    .WithEnvironment("ACCEPT_EULA", "Y")
    .WithNetwork(network)
    .WithDatabase(new MsSqlTestcontainerConfiguration()
    {
        Password = "UltraSecret92120!!"
    })
    .Build();
```

Komt dit jou **bekend voor?**

"Je organisatie heeft een succesvol product in de markt staan. Echter de releases worden steeds complexer en kosten meer inspanning om uit te brengen. Het team heeft steeds meer bezwaren als er features worden aangevraagd? Dit terwijl de markt steeds sneller wordt. Je klanten verwachten meer updates, betere ondersteuning van jouw organisatie en natuurlijk moet je product voldoen aan de wetgeving. Al deze verwachtingen van klanten worden jouw verplichtingen. En die verplichtingen worden steeds moeilijker waar te maken. Je hebt het gevoel dat het ontwikkelteam niet meer voldoet aan de verwachtingen van klanten. Je hebt het gevoel dat het ontwikkelteam jouw ambities remt? Je hoofd zegt dat er iets moeten veranderen in het ontwikkelteam..."

Auteur: Martijn Broos

Maar is dat wel zo, ligt het probleem van de organisatie wel alleen bij het ontwikkelteam?
Het kan zijn dat het team niet optimaal functioneert, echter dat is vaak niet het geval. Er zijn meerdere factoren die een rol spelen om op een toekomstbestendige manier software te maken. In dit artikel benoem ik 5 punten in software ontwikkeling die in mijn beleving belangrijk zijn om toekomstbestendig te zijn:

> Strategie en roadmap
> Software ontwikkelprocessen
> Het ontwikkelteam
> Best practises
> Dev(Sec)Ops

Strategie en roadmap

Als buitenstaanders merken we vaak op dat het ontwikkelteam wacht op input. Het ontwikkelteam weet niet wat er van hen verwacht wordt. Welke visie heeft de organisatie en ook welke strategie wordt toegepast?
Een herkenbaar fenomeen is het toevoegen (en blijven toevoegen) van veel features aan een product. Maar zit de markt hier wel op te wachten? Wat is de toegevoegde waarde van de features? Je kan echter jezelf beter afvragen, "Welke features maken het verschil in de markt?"

Een gedegen strategie, gebaseerd op de marktwensen, zorgt ervoor dat het ontwikkelteam doelgericht waardevolle features toevoegt in plaats van code kloppen. Heel simpel gezegd; Een product met alleen essentiële features is eenvoudiger te onderhouden dan een product dat vol zit met ongebruikte features.
Goede afstemming tussen stakeholders en het ontwikkelteam zorgt ervoor dat er een solide roadmap ontstaat met prioriteiten. De roadmap geeft een duidelijke focus en doel weer. Ontwikkelaars voelen zich over het algemeen beter en meer op hun gemak bij een duidelijk doel. Verwachtingen zijn duidelijk. Op basis van deze verwachtingen kan een team commitment tonen. Het team zal verantwoording nemen over het ontwikkelproces. Een duidelijke strategie en roadmap zorgt dat het ontwikkelteam autonoom wordt.

Software ontwikkelprocessen

De meeste organisaties hebben de laatste jaren wel vormen van Scrum ingevoerd. Helaas is dat voor veel organisaties een eindstation in plaats van een beginpunt om wendbaarder te worden. Mijn ervaring is dat er niet één proces is dat het antwoord op alle vragen biedt maar dat in de praktijk gecombineerde best practices uit Scrum, Kanban en Extreme Programming, de beste resultaten boeken.
De vraag die overblijft is, "Past het ontwikkelproces wel bij de rest van de organisatie?" Het invoeren van

Er is niet één proces dat het antwoord op alle vragen biedt, een goede mix helpt wel

ontwikkelprocessen kan conflicten opleveren met andere afdelingen in de organisatie. 2 voorbeelden waar frictie kan ontstaan door verschillende inzichten;

- Een organisatie is project gedreven met klant specifieke implementaties van een standaard product. In een dergelijke organisatie zal de commerciële afdeling vaak een vaste prijs, vastgestelde functionaliteit en deadline met de klant afspreken. Dit is tegenstrijdig aan Scrum omdat het team hier geen verantwoording voor kan nemen. Binnen Scrum staat de eigenaarschap van het ontwikkelteam centraal. Hoe kan een ontwikkelteam eigenaarschap nemen als zowel de prijs, tijd en functionaliteit is vastgelegd en wordt opgedrongen door de commerciële afdeling?
- Een organisatie heeft een support afdeling die volgens de ITIL-methodiek werkt. ITIL heeft een ander doel voor ogen dan Scrum. In essentie heeft ITIL als doel om verandering zoveel mogelijk te beperken en te beheersen. Dit wordt gedaan door processen effectief en efficiënt in te richten om zo snel mogelijk klantvragen te kunnen beantwoorden. Scrum is vooral een empirisch werkmethode wat verandering omarmt. Scrum anticipeert op voortschrijdend inzicht en verwachte wijzigingen in de toekomst.

Als organisatie moet je hier wat mee doen. Zoeken naar het juiste ontwikkelproces, dat het beste past bij de markt en je eigen organisatie. Dit hoeft niet perse het meest efficiënte ontwikkelproces te zijn. Een proces dat aansluit bij de organisatie zorgt ervoor dat een ontwikkelaar focus heeft en zo optimaal mogelijk kan

werken. Het resultaat is een grotere impact en betere kwaliteit.

Het ontwikkelteam

Een goed team hebben is één ding, een goed team behouden is iets anders. Organisaties zijn trots dat medewerkers lang bij het bedrijf willen werken.
En dan komt "de maar". Software ontwikkeling en -talen veranderen continu en zullen de komende tijd nog meer gaan veranderen. Kennis die 10 jaar geleden is opgebouwd, is niet meer courant. Wij geloven dat investeren in software ontwikkelaars essentieel is om bij te blijven met de ontwikkeling in de markt.
Dit betekent een gedegen ontwikkelplan, goede tools en tijd om met innovatie en zelfstudie bezig te zijn. Zorg ervoor dat de ontwikkelaars op de hoogte zijn van de architectuurveranderingen, huidig geldende standaarden en ontwikkel filosofieën zoals SOLID, YAGNI en KISS; Allemaal filosofieën die zorgen dat software anders, beheersbaarder, wordt opgebouwd.
Naast technische skills wordt er steeds meer van een ontwikkelaar gevraagd. Denk eens aan de beveiligingseisen die de AVG tegenwoordig stelt; Er is een gedachteverandering dat security, wat een beheer/support probleem was, nu steeds vaker op het bordje van de softwareontwikkelaar terecht komt. Hier zal een ontwikkelaar kennis en kunde voor moeten opbouwen om de materie te begrijpen.
Een goed ontwikkelplan zorgt er voor dat medewerkers zich gehoord en gewaardeerd voelen. Ze zullen langer bij jouw organisatie willen blijven. Dit is een groot voordeel in de huidige markt waar iedereen schreeuwt om kwalitatief personeel.
Naast de individuele eigenschappen van alle teamleden zijn ook de grootte en de samenstelling van het team van belang. Zijn alle noodzakelijke rollen voldoende beschikbaar in het team? Dat betekent niet alleen dat de noodzakelijke rol in het team aanwezig is, maar ook beschikbaar is op het moment dat een ander teamlid die rol nodig heeft. Is het team wel groot genoeg om de doelstellingen van de organisatie te halen of is het team misschien zelfs te groot?
Een bekend fenomeen is de manager die bij een lage impact van het ontwikkelteam meer teamleden gaat toevoegen. Meer teamleden toevoegen wil niet automatisch zeggen dat er ook meer werk verzet wordt. Er zijn afhankelijkheden van elkaar en je zult moeten wachten in sommige situaties. Meer teamleden in je team betekent ook dat je overhead toeneemt. Je zal meer communicatie nodig hebben om alle teamleden geïnformeerd te houden. Teveel teamleden toevoegen kan de impact zelfs verlagen in plaats van verhogen.

Best practices (tooling en training)

Voor elk probleem bestaan er legio oplossingen. Dat was altijd al zo, maar als je kijkt naar het aantal tools en platformen dat ontwikkelaars

vandaag de dag kunnen gebruiken is deze uitspraak meer dan ooit actueel. Het is onmogelijk om expert te zijn in elke tool, des te belangrijker om aandacht te geven aan het ontwikkelen van best practices die de tand des tijds kunnen doorstaan en tool onafhankelijk zijn.
Clean coding en het opzetten van de juiste architectuur, helpt om onderhoudbare software op te zetten ongeacht de gekozen tools. Best practises kun je echter alleen selecteren en implementeren als je in het voorgaande punt ook een goed persoonlijk ontwikkelplan hebt. Hoe is een ontwikkelaar in staat om de juiste best practise te kiezen als de ontwikkelaar niet weet dat deze er is?
Geef als organisatie ruimte om te experimenteren; om uit te zoeken hoe een nieuwe techniek werkt, om op basis van deze ervaring gecombineerd met parate kennis een best practise uit te werken. Best practises zorgen er voor dat ze algemeen toepasbaar zijn.

BIO

Martijn Broos

Martijn Broos, Technical Lead Consultant bij Bergler “Ik help graag software architectuur en ontwikkelprocessen te optimaliseren”

Nieuwe collega’s zullen deze best practises herkennen en sneller ingewerkt zijn. Vraag jezelf af, hoeveel tijd heeft een nieuwe ontwikkelaar nodig om productief in jouw organisatie te zijn? Als dat langer is dan een maand, dan is dit een indicatie dat er niet future proof ontwikkelt wordt op dit moment.

Dev[Sec]Ops

Effectief software ontwikkelen gaat niet alleen over code kloppen. Vanaf het eerste idee, tot het moment dat de software draait, moet alles werken als een goed lopend uurwerk. Elk tandwieltje grijpt in het volgende tandwieltje. Hiervoor is het van belang om development, operations en security goed te integreren.
Dev(Sec)Ops is niet alleen een term; het is integraal onderdeel van het ontwikkelproces en een way of working. Het proces richt zich op het ontwikkelen, beveiligen en operationeel houden van een applicatie door 1 team. Hierbij is het belangrijk dat alle kennis en kunde binnen het team aanwezig is om alle drie de facetten te kunnen beheersen.
Het proces staaft op een aantal regels of principes:

> Het “shift left” principe. Shift left wil zeggen dat je steeds vroeger in het proces gefocused bent op issues. Issues kunnen zijn op basis van code kwaliteit, beveiliging maar ook o.a. performance. Hoe sneller de organisatie issues identificeert en meeneemt in de oplossing, des te minder effort zal het oplossen kosten. “Threadmodelling” en “Secure by Design” zijn methodieken om potentiële beveiliging issues inzichtelijk te maken.
> Alles wat meer dan 1 keer wordt uitgevoerd, moet je automatiseren. Dit gaat niet alleen over deployment maar ook over processen in de ontwikkeling. Code kwaliteit checken door middel van unit-testen of static code analysis zijn mogelijkheden om te automatiseren. Hoe minder handmatige handelingen je uitvoert, hoe minder fouten er gemaakt kunnen worden.
> Alles wat complex is of risicovol is, moet je automatiseren. Is je deployment lastig en vraagt het veel stappen? Dan is automatiseren de oplossing. Het voorkomt dat de mens fouten maakt in het proces.
> Het proces is traceerbaar en beheersbaar. Vanaf het moment dat er een idee is, tot het moment dat het idee in productie geplaatst is, hoort het proces traceerbaar te zijn. Backlog items die gekoppeld zijn aan versies, bugs die gekoppeld zijn aan backlog items, testresultaten gekoppeld aan versies en uiteindelijk ook versies gekoppeld aan deployments.

Dev(Sec)Ops vraagt initieel meer van je ontwikkelteam maar zal uiteindelijk de kwaliteit van de code verbeteren en zorgen dat het ontwikkelteam meer impact kan maken.
De allereerste paragraaf is een extreem voorbeeld. En toch zijn er organisaties die zich erin herkennen. Het is slechts één voorbeeld van wat er allemaal minder goed kan gaan tijdens software ontwikkeling. Als je naar future proof software development kijkt, dan doet de hele organisatie mee. Het ontwikkelteam pakt Dev(Sec)Ops op. De scrummaster kan zich bezig houden met het ontwikkelproces en de manager van het team zal in de organisatie moeten gaan kijken hoe het ontwikkelplan wordt vormgegeven. Wellicht zal een directeur moeten regelen dat er tijd wordt vrijgemaakt voor experimenteren. De product owner kan samen met de product manager en stakeholders zorgen voor een productvisie en roadmap. Faalt 1 radartje in de organisatie, dan kan dat gevolgen hebben voor de impact die het ontwikkelteam kan maken.
Organisatie komt eigenlijk van het werkwoord organiseren. Zaken regelen maar vooral samen werken om een gemeenschappelijk doel te bereiken. ●

Cybersecurity in DevOps

Auteur: Raymond Jetten

Als DevOps engineer ben jij verantwoordelijk voor het ontwikkelen en onderhouden van de IT-systemen van je opdrachtgever. Alsof die taken nog niet genoeg zijn, is er ineens een cybersecurity incident op een van de systemen die onder jouw verantwoordelijkheid vallen. Nu blijkt dat je niet alleen verantwoordelijk bent voor ontwikkeling en onderhoud, je bent ook nog eens de eerste verdedigingslinie. Hoe kan het dat je niet alleen onderhoud en ontwikkeling moet doen, maar ook nog eens de beveiliging van je systemen? Je bent niet de enige die met dit probleem worstelt. Veel DevOps teams zitten met eenzelfde vraag of zullen hierop enig moment tegenaan lopen, simpelweg omdat ze vroeg of laat met een cybersecurity incident geconfronteerd worden. De wereld is namelijk aan het veranderen: landen voeren digitaal oorlog, criminelen kunnen online enorm cashen en iedereen kan een (niet zo ethische) hacker zijn. Het is geen vraag meer óf je cybersecurity incidenten kunt verwachten, maar wanneer je ze moet verwachten. Dit artikel helpt jou als DevOps engineer om je voor te bereiden op de onvermijdelijke cybersecurity incidenten.

Scope, inventariseer en leer je systemen kennen

"Je kunt je systemen niet verdedigen, als je niet weet wat je precies hebt." Voordat je aan de slag kunt, zul je eerst moeten weten wat je scope is. Het maakt niet uit of je gehele infrastructuur in een cloud omgeving hangt of in een serverpark op kantoor staat. Begin met het afbakenen van de scope van de infrastructuur die onder jouw (of je teams) verantwoordelijkheid valt. Zorg ook dat iedereen, inclusief je baas, het eens is met deze scope. Vermijd grijze gebieden waarbij meerdere partijen (of niemand!) verantwoordelijk zijn. Ben je alleen verantwoordelijk voor de applicaties die jij en je team ontwikkelen en beheren, of wordt er ook van je verwacht dat je de Macbook verdedigd waar Harold van sales op werkt, en 's avonds mee naar huis neemt om stiekem films mee te torrenten? Zodra je scope duidelijk is kan je gaan inventariseren. Waarschijnlijk kan je 80% uit je hoofd benoemen, maar hoe zit het met die hele specifieke details of wanneer je nieuwe collega met twee maanden ervaring een incident moet afhandelen? Zorg ervoor dat je een up-to-date overzicht van al je assets tevoorschijn kunt halen zodra je het nodig hebt. Dus niet alleen fysieke assets, maar ook virtuele assets, software in gebruik, data flows, enz. Voor kleine omgevingen kom je misschien nog weg met een Excel Sheet, maar voor grotere en meer dynamische omgevingen ben je beter af met een van de vele beschikbare tools voor assetmanagement die op de markt te krijgen zijn (gratis of betaald). Zorg ervoor dat je dit overzicht ook beschikbaar hebt als je productieomgeving niet beschikbaar is en dat het zo compleet mogelijk is.

Nu je weet wat je moet beschermen en hoe het eruit ziet, is het tijd om te leren hoe het écht werkt. Natuurlijk weet je wat het product doet, maar weet je ook wat er onder de spreekwoordelijke motorkap gebeurt? Gebruik je open-source container images die telemetrie naar de uitgever sturen? Is de spring-boot applicatie die op poort 8080 luistert, ook echt de enige service op je virtuele machine die verbindingen accepteert?

Wat je ook vindt in deze eerste stappen, documenteer het. Het maakt niet uit of de resultaten precies zijn zoals je verwacht of dat je wellicht wat verras-

> "Je kunt je systemen niet verdedigen, als je niet weet wat je precies hebt."

> “Als je de statische content van je website gaat air-gappen kun je uitdagingen verwachten.”

singen gevonden hebt. De informatie die je hier verzameld gaat jou en je collega’s enorm helpen in het detecteren van mogelijke incidenten, maar ook in het analyseren en oplossen ervan.

Bescherm je systemen

“Als je de statische content van je website gaat air-gappen kun je uitdagingen verwachten.”
Wellicht ben je geschrokken van wat je in de scoping-fase gevonden hebt en wil je zo snel mogelijk zorgen voor een betere beveiliging. Je kunt natuurlijk delen van je infrastructuur loskoppelen van het internet (air-gappen), multi-factor authentication afdwingen en super complexe wachtwoorden die elke week wisselen verplichten, maar is dat écht de beste manier om de server die statische website content levert te beveiligen? Waarschijnlijk niet.
Goede beveiliging begint bij het leren kennen van jezelf en je tegenstanders. Is de organisatie of het bedrijf waarvoor je werkt een grote multinational die diensten voor kritische infrastructuur levert, of een mkb’er die kinderspeelgoed verkoopt? Beide hebben te maken met een heel verschillend dreigingsbeeld (threat-environment). De multinational zal waarschijnlijk opgewassen moeten zijn tegen Advanced Persistent Threats (APT’s, zeer geavanceerde hackersgroepen) die uit zijn op maatschappelijke schade, de speelgoedverkoper zal vooral te maken hebben met criminelen die snel willen scoren. Ga voordat je zelf aan de slag gaat op zoek naar beschikbare informatie. Sommige organisaties hebben interne threat-environments of threat-models beschikbaar die zijn gemaakt door de security afdeling, anderen kunnen wellicht terecht bij een CERT (Computer Emergency Response Team) dat een specifieke sector bedient. Zo is er bijvoorbeeld een sectoraal CERT voor de zorg.
Als je weet wie je tegenover je hebt, kun je aan de slag gaan met de implementatie van beschermende maatregelen die relevant zijn voor jouw specifieke situatie. Uiteraard zijn er een aantal best-practices. Zo kunnen lange wachtwoorden, encryptie van data en multi-factor authentication nooit kwaad. Wat daarop aanvullend nog relevant is zul je zelf moeten bepalen. Denk bijvoorbeeld eens na over hoe

> “Het incident was duidelijk vindbaar. Gewoon, in de logs.”

kwetsbaarheden in je systeem tot een minimum kunnen worden beperkt en snel kunnen worden verholpen.
Hoe dan ook, het is belangrijk om een goede balans te vinden tussen de gebruiksvriendelijkheid van het systeem en de security. Een overdreven voorbeeld is het air-gappen van de statische content van je website. Het neemt vrijwel alle reguliere manieren om je website aan te vallen weg, maar maakt normaal gebruik ervan ook onmogelijk. Maak je risico’s inzichtelijk en zoek een goede balans tussen risico-acceptatie en risico-mitigatie.

Er zijn frameworks en standaarden die je hierbij kunnen helpen zoals de ISO 27000/27001, het NIST Cybersecurity Framework en het NIST Risk Management Framework.

Afwijkingen detecteren

“Het incident was duidelijk vindbaar. Gewoon, in de logs.”
De voorbereiding is klaar en beschermende maatregelen zijn genomen, dus nu kan het verdedigen echt beginnen. Aangezien je DevOps team geen volledig Security Operations Center (SOC) is, zul je zo efficiënt mogelijk moeten zijn. De meeste DevOps teams hebben al een log-verzameling tool in gebruik (DataDog, ElasticSearch, etc.). Dit soort tools richten zich steeds meer op security, dus om de eerste stappen op het gebied van detectie te zetten hoef je vaak maar een paar keer te klikken om de relevante features (SIEM, Security Information & Event Management) en logs aan te zetten. De sleutel tot succes is in dit geval echter: Less is more. Ga selectief te werk met het verzamelen van security logs. Je wil namelijk voorkomen dat je een overdaad aan informatie krijgt waardoor incidenten in de logs staan, maar niemand ze opmerkt.
Houd ook in deze stap weer rekening met het dreigingsbeeld dat op jouw situatie van toepassing is. Verzamel vooral logs die je helpen om dreigingen die op jouw situatie van toepassing zijn te detecteren. Heb je vooral te maken met generieke malware? Die is vaak snel te herkennen in DNS logs. Maak je je zorgen om ransomware? Ga dan aan de slag met Endpoint Detectie & Response (EDR) software. Ben je een

“Hoop op het beste, plan op het slechtste.”

mogelijk doelwit voor APT’s? Zorg dan voor up-to-date Intrusion Detection/ Protection Systemen (IDS/IPS) en houd de logs daarvan scherp in de gaten. Uiteraard heeft het weinig zin om security gerelateerde logs alleen te verzamelen, je zult er ook iets mee moeten doen. Zorg ervoor dat de oplossing die je gebruikt om je logs te verzamelen jou een alert geeft op het moment dat zich een situatie voordoet die je wilt voorkomen. Bijvoorbeeld: EDR-software die een ransomware infectie voorkomt, DNS query’s naar top-level domains die je niet verwacht, of services die verkeer ontvangen op onverwachte poorten. Voor het makkelijk detecteren van dit soort events zijn SIEM oplossingen ontwikkeld. Als je al een logoplossing in gebruik hebt met SIEM functionaliteit, gebruik die dan. Natuurlijk kun je niet 24/7 naar de output van je monitoring kijken. Zorg er daarom voor dat je de tool die je hiervoor gebruikt, jou op de hoogte stelt van specifieke alerts. Of dat nou een push-melding, Jira ticket of Slack melding is maakt niet uit. Dit is de fase waarin jij als first responder kunt shinen! Wil je nog iets verder gaan? Neem zo nu en dan even de tijd om net wat dieper in je logs en systemen te duiken. Denk eens na over hoe jij jouw eigen systeem zou aanvallen en ga vervolgens na of deze aanvallen ook daadwerkelijk plaatsvinden. Dit concept heet Threat Hunting en wordt normaal gesproken door level 2 of 3 cybersecurity analisten gedaan. Het is echter een geweldige manier om meer te leren over cybersecurity en je systemen beter te leren kennen.

Reageren op een incident

“Hoop op het beste, plan op het slechtste.”

Wat als je concludeert dat we daadwerkelijk met een cybersecurity incident te maken hebben? Ga je getroffen containers opnieuw deployen en hopen dat het niet nog eens gebeurt? Het is belangrijk om een plan klaar te hebben voor de momenten waarop het mis gaat. Het zorgt ervoor dat je een probleem gestructureerd aanpakt en de chaos die een incident met zich meebrengt de baas kunt blijven. Bedenk een aantal (realistische) scenario’s op basis van jouw threat environment. Wat gaat er mogelijk mis en wie moet er dan op reageren? Kun je het betreffende scenario zelf afhandelen of heb je ondersteuning nodig van een incident response team? Wie moet je informeren, hoe herstel je je systemen en hoe voorkom je dat een incident terugkeert net nadat je het hebt opgelost? Het zijn allemaal vraagstukken waar je vooraf over nagedacht moet hebben om chaos tijdens een incident te voorkomen.
Je kunt online veel templates en voorbeelden vinden voor het maken van een incident response en recovery plan. Geen van deze voorbeelden zijn echter allesomvattend en specifiek voor jouw situatie geschreven. Het is daarom belangrijk om je plannen zo nu en dan te testen. Zet een component met sleutelrol uit in je testomgeving en behandel het alsof het onderdeel is van een cybersecurity incident. Hoe reageert je team? Doet het plan wat het moet doen? Wat kan er beter?
Als laatste zul je ervoor moeten zorgen dat je plannen mee veranderen met je infrastructuur en systemen. Het updaten van documentatie is vaak onderdeel van de definition of done van development teams. Zorg ervoor dat incident response plannen bijwerken daar ook onderdeel van is. Als er nieuwe componenten worden toegevoegd, zullen je plannen daar rekening mee moeten houden.

Conclusie

Zodra je het bovenstaande meeneemt in je DevOps werkzaamheden, zul je aanmerkelijk beter in staat zijn om cybersecurity incidenten te detecteren en erop te reageren. Maak daarbij gebruik van procedures en tools die je organisatie al heeft geïmplementeerd. Het verdedigen van IT-systemen tegen cyberaanvallen is namelijk een teamsport, waarbij de verdedigers alle mogelijke manieren om aangevallen te worden moeten verdedigen, terwijl de aanvallers genoeg hebben aan een enkele treffer. Maak het onderwerp daarom bespreekbaar binnen je team, management en de rest van je organisatie en ben realistisch in wat je wilt bereiken. Kleine, structurele stapjes zijn ontzettend waardevol en beter vol te houden dan in één klap alles goed willen doen. ●

BIO

Raymond Jetten

Raymond Jetten is cybersecurity engineer bij Team Rockstars IT. Volgens Raymond is cybersecurity een onmisbaar onderdeel van moderne softwareontwikkeling.


FUTURE
TECH 2023
THE IT CONFERENCE FOR
MICROSOFT TECHNOLOGIES
March 15 2023, Jaarbeurs Utrecht
Tickets: www.futuretech.nl
Are you member of the SDN? Get your
free ticket with code: SDNMember

Keynote: Tim Huckaby - Ethics of AI - Sessie in the Progress - 13:45

Artificial Intelligence (AI) has the processing capabilities of speed, scale, and capacity far beyond that of humans. As computers become more powerful, Machine learning (ML) the "brains" behind AI become exponentially more powerful.

AI's bold promise to improve efficiency, lower costs, and fast-track R&D has been tempered with the fear that many of these complex, obscure systems may do more societal harm than economic good.

With virtually no government oversight, companies are using AI to make determinations about health and medicine, employment, insurance, creditworthiness, and even criminal justice without any form of requirement to ensure compliance to privacy law nor the inclusion of structural biases.

> Can that power always be trusted to be fair and neutral?
> Can that power be trusted to protect your privacy within civic, state and federal laws worldwide?
> Can the government hysteria of removing facial recognition technology from authorities like the police make me less safe against terrorism?
> When Facebook or Google's Photos service uses AI on my personal photos to identify people, locations, objects and scenes and consequently targets internet advertising at me based on that AI is it an invasion of my privacy?

AI has already proven for years that it is saving lives. Not that it has to potential to save lives; it is already saving lives and has been for years. From cancer research to medical diagnosis to safety and security to covid-19 detection and everything in between; AI is saving lives.

Yet we have technology industry pillars like Google, Amazon, and Facebook, leaders AI, who's revenue, and consequently shareholder value is rooted in internet advertising. Companies who are using AI to track your activities on the internet to detailed granularity which produces influenceable advertising that is simply scary. And these companies have basically been doing this unimpeded for years. And then there is Microsoft, arguably the leader in AI in terms of sheer R&D, with revenue and shareholder value mainly based in cloud consumption who is pushing federal governments worldwide hard for AI regulation.

It is years beyond the right time to talk about the nearly boundless landscape of artificial intelligence. In many ways, AI is just as much a new frontier for ethics and risk assessment as it is for emerging technology. Join Tim Huckaby in the discussion; a demo heavy keynote which will elaborate the power, the risk, the ethical dilemmas we are currently facing in AI and that we will face for many years to come.

Microsoft

Expedition 11:00 - François Bouteruche - REST, gRPC, SignalR and GraphQL for .NET developers. Which is right for your use case?

REST, gRPC, SignalR and GraphQL. As .NET developers, we hear a lot about those four technologies to build API. However, it is not always crystal clear how to decide the one to pick for a use case. REST is a software architectural style. gRPC is a cross-platform open source Remote Procedure Call (RPC) framework. SignalR is a free and open-source software library. GraphQL is an open-source data query and manipulation language for APIs.

In this talk, we discuss their core concepts and the problems they are trying to solve. We showcase how to quickly bootstrap an API in C# with each one of them. Finally, we propose a decision-tree to help you decide the right one for your use case.

aws

Mission 1 11:00 - Barbara Forbes - GitHub Actions to deploy to Azure: Take it to the next level

A basic deployment to Azure from GitHub Actions can be set up in 10 minutes. And we will do that, to get the basics down. But let's take it a step further. How can you make sure a workflow runs as efficient and flexible as possible? What options do you have to make a connection? Can different Bicep templates interact with each other? How do you work with stages? How do you make the workflow act differently based on branch or pull request label? Can you implement manual approvals?
In this session we will start with the basic workflow and walk through these great options that GitHub has to offer to get your workflow to work for you.

Progress 15:45 - Geert van der Cruijsen & Thijs Limmen - Is ChatGPT a Better Software Engineer Than Me?

Artificial intelligence (AI) and machine learning have the potential to revolutionize the field of software engineering. One example of this is ChatGPT, a large language model developed by OpenAI.

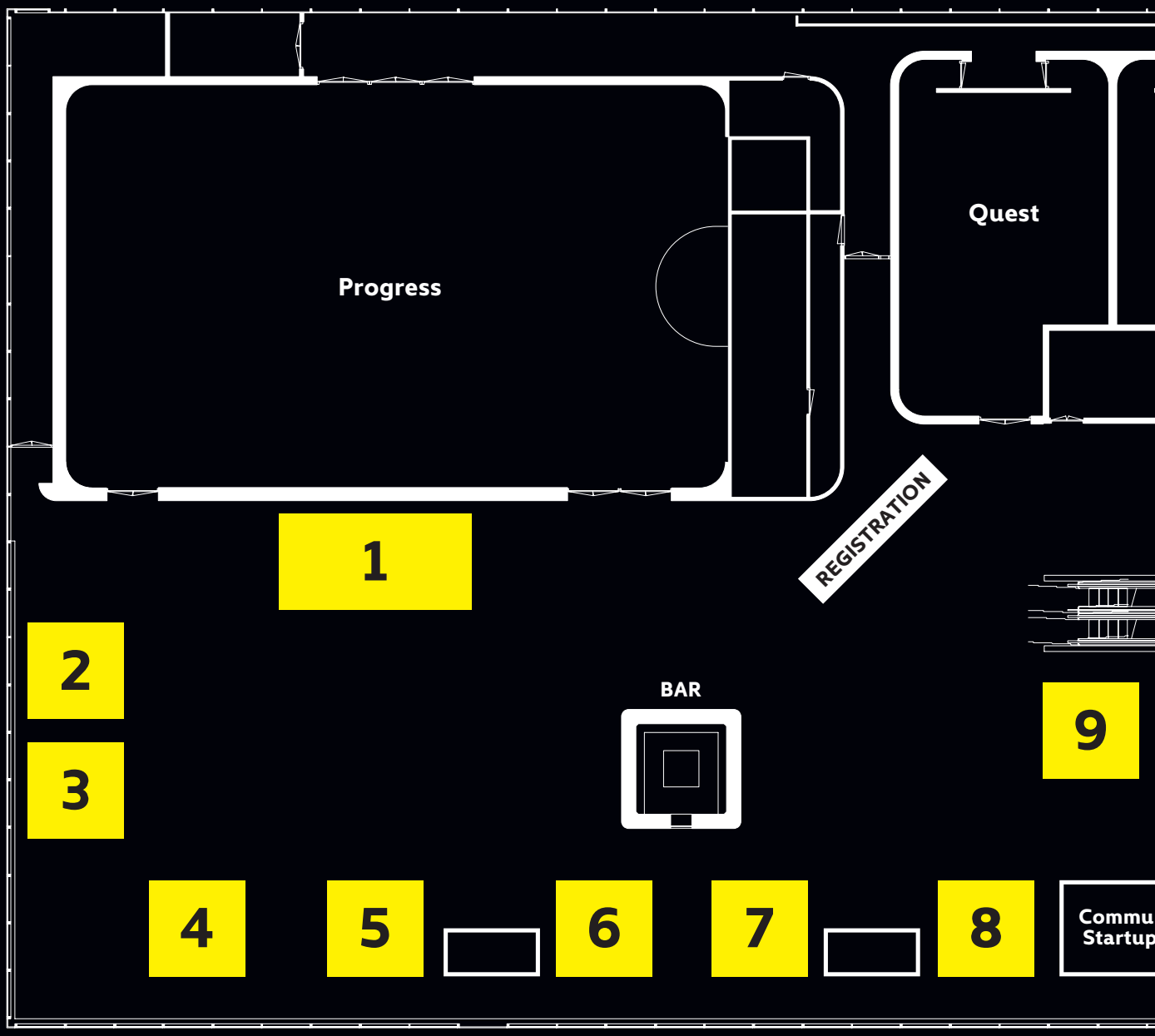
In this session, we will explore the capabilities of ChatGPT as a software engineer, including its ability to write code, debug, and troubleshoot issues. We will also discuss the limitations and challenges of using AI in software engineering, and consider the implications for the future of the field.

Whether you are a software engineer yourself or simply interested in the role of AI in technology, this session will provide insights and thought-provoking discussions on the intersection of AI and software engineering.

Mission 1 14:30 - Maarten Goet - 52 mins from initial access to ransomware -- is your defensive team ready?

Does your defensive team detect and respond fast enough to contain the adversary before they accomplish their mission? How do you know? Come join 15-year Microsoft MVP & RD Maarten Goet and learn how Microsoft Defender 365 and Microsoft Sentinel can help detect command-and-control platform Cobalt Strike, nasty exploits and much more, in this real-world case where an attacker went from Zero to Ransomware in just 52 minutes. This is a session no security admin should miss!

Floorplan Future Tech 2023



Netcompany	1	Chipsoft	4	Ordina	7
Bergler	2	Vitas	5	Achmea	8
SDN cast	3	Betabit	6	4DotNet	9

*Subject to change


Expedition
Mission 1
Mission 2
lift
Workshops
pubquiz
nity &
plein

Timetable Future Tech 2023

Room/Time	Progress	Mission 1	
10:00 - 10:10	Official Opening Future Tech 2023		
10.10-10.40	**Keynote: Alexander Colliander Hansen**		
10.40-11:00	**Transit zone: Coffee break**		
11.00-11.45	Pub crawling in Orleans: exploring the Actor model **Sander Molenkamp**	GitHub Actions to deploy to Azure: Take it to the next level **Barbara Forbes**	
12.00-12.45	An introduction to decentralized identity **Stefan van der Wiele**	Azure Functions Isolated, the same but different **Oscar van Tol**	
12.45-13.45	**Transit zone: Lunchbuffet**		
13.45-14.30	**Keynote: Ethics of AI – Tim Huckaby**		
14.30-15.15	Real-Time Connected Apps with .NET MAUI, Blazor and SignalR **Gerald Versluis**	52 mins from initial access to ransomware - is your defensive team ready? **Maarten Goet**	
15.15-15.45	**Transit zone: Coffee & tea break**		
15.45-16.30	Is ChatGPT a Better Software Engineer Than Me? **Geert van der Cruijsen & Thijs Limmen**	The intersection point of Low-code vs Pro-Code projects with regards to return on investment **Iona Varga**	
16.45-17:30	Explaining your AI models - Why & How **Sammy Deprez**	Human vs AI: How to ship secure code with GitHub CoPilot **Joseph Katsioloudes**	
17.30-18.30	**Transit zone: Drinks & Music**		

*Subject to change

	Mission 2	Expedition	Quest	Workshop area
	How to really work as a team - from sleepy to strong pair programming **Stacy Cashmore**	REST, gRPC, SignalR and GraphQL for .NET developers. Which is right for your use case? **François Bouteruche**	Building and Running .NET Apps on the Cloud- Choose your own adventure **Isaac Levin**	Creating APIs on Steroids with gRPC **Roland Guijt**
	Building a Responsible AI strategy **Sammy Deprez**	Escaping the Legacy Trap - a proven method for application modernization **Michael Coté**	Finding the Root Cause. Distributed Tracing in .NET and Azure **Annejan Barelds**	
	The Hitchhiker's Guide to (Microsoft) Certification **Michiel Hamers**			
	What Has Public Speaking Ever Done For Us? **Stacy Cashmore**			
	Creating a quantum algorithm using Microsoft Q# **Johnny Hooyberghs**	What if all the boring stuff is already done? **Christiaan van 't Hoft & Thijs Vorselen**	Why Wasm is the perfect runtime for server-side applications. **Eelco Koster**	How to build a web3-app: from design to smart contract **Jessy The**
	Using WebAssembly to run, extend, and secure your .NET application **Niels Tanis**	Design for costs, four pillars of economic cloud design **Michiel Hamers & Twan Koot**	Het spanningsveld tussen theorie en praktijk op het gebied van Kunstmatige Intelligentie in de zorg. **Marc van Somberg & Maarten van Rood**	
	Blazor: Blazing into the Future of Web Development **Roland Guijt**	Innovatie binnen Achmea **Ignace Konig**	O shit... how do I get rid of my legacy code **Menno Jongerius & Rik Lammers**	Pubquiz **powered by XPRTZ**

Make a **difference**

At Netcompany we believe that our people can make a real difference to our societies, businesses and the environment. Our culture is open and driven by collaboration. Our flat organizational structure allows for greater agility and a pragmatic approach focused on delivering optimal results.

Technology is at the centre of everything we do at Netcompany. What guides us is a core belief that we can use it to help make a real difference to our common future. We challenge old business models providing new disruptive services. We do this for more than hundreds of clients in the private and public sector. Read more about our projects and the value they create in society on our website: **www.netcompany.com**

We are Netcompany

Netcompany was founded in 2000 and has its headquarters in Copenhagen, Denmark. Today, we are an international company with more than 7,400 employees working from 17 countries. We serve a wide array of customers in their digital transformation journeys across Europe. In the Netherlands we are located in Delft with clients such as Studielink, Rijkswaterstaat and the Ministry of Foreign Affairs.

Netcompany is a pure-play IT-services company delivering business-critical strategic IT projects. Our goal is to accelerate customers' digital transformation through digital platforms, core systems and infrastructure services.

Our values

It takes talent to apply the opportunities technology gives us at Netcompany and Netcompany invests in talent like no one else. Every project is led by project managers with strong IT backgrounds, who have the experience to guide the team and to ensure quality in all deliverables while knowing what responsibility to delegate. Our ambitions for the future are big: we aim to be the leading digital challenger in Northern Europe.

Be an expert and lead the way

Are you ready to take on greater responsibility and develop your technical and managerial skills? Do you want to achieve more as a leader or senior in the IT industry? At Netcompany, you will continue to develop yourself, wherever you are in your career. You will be challenged but supported. You will learn from your peers, our myriad projects and an extensive development program, including our renewed Netcompany Academy.

Meet Netcompany at Future Tech!

Want to know more? We are eager to meet you and share more about what we do and how we can make a difference together.

Join our parallel session: What if all the boring stuff is already done?

Come and join our colleagues Christiaan van 't Hoft and Thijs Vorselen, two of our managing architects during their session at Future Tech. Can we make your life as a developer easier? They will give you an in depth look at a production proven .net codebase that includes all the code you normally produce in the hectic prebuild phase. This codebase adheres to the clean architecture principles and implements Command-Query Separation with the mediator pattern.

netcompany

FUTURE TECH 2023

THE IT CONFERENCE FOR MICROSOFT TECHNOLOGIES

Future Tech thanks her partners

MAIN SPONSOR

netcompany

PARTNERS

4dotnet

achmea

BERGLER SOFTWARE SOLUTIONS

betabit
samen bijzonder maken;

ChipSoft

ORDINA

VITAS

XPRTZ

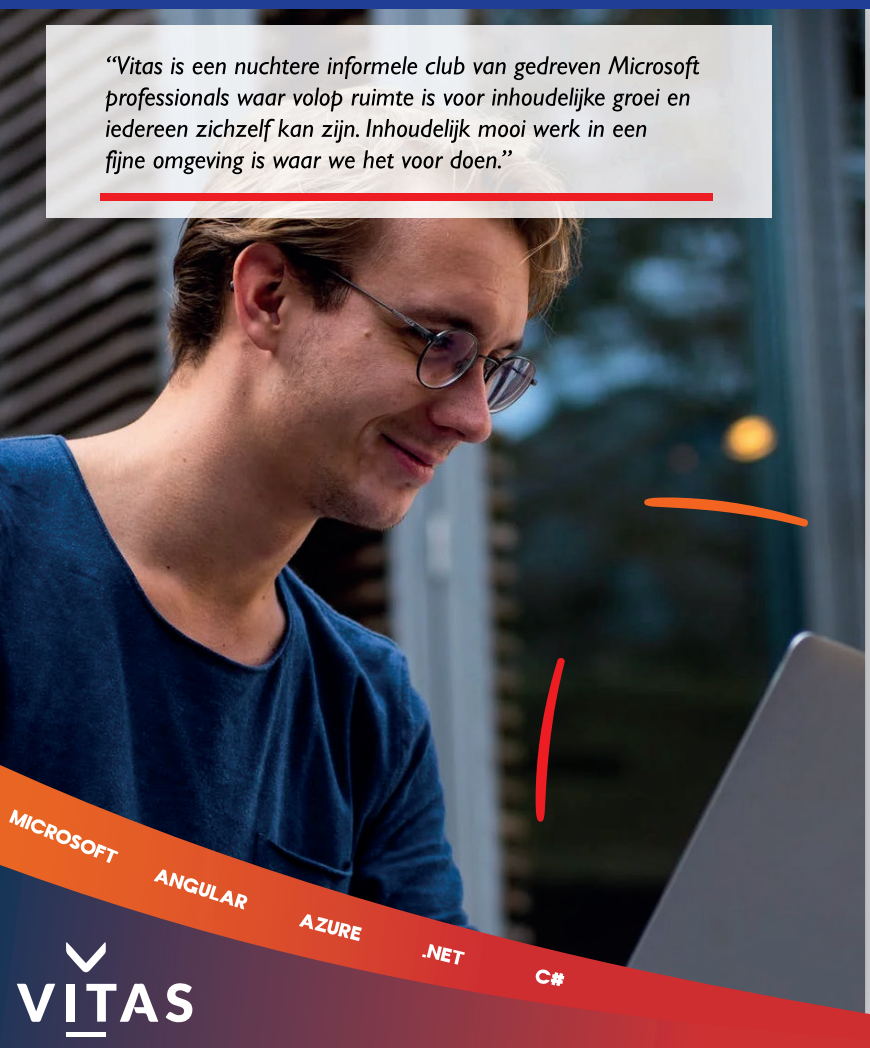
4dotnet

Met een team van 50 .NET developers delen we onderling kennis & ondernemen we de tofste uitjes!



Onze diensten
- Consultancy
- Detachering
- Trainingen

Scan de QR code en bezoek onze website:



"Vitas is een nuchtere informele club van gedreven Microsoft professionals waar volop ruimte is voor inhoudelijke groei en iedereen zichzelf kan zijn. Inhoudelijk mooi werk in een fijne omgeving is waar we het voor doen."

VITAS

MICROSOFT ENTHUSIASTS

We houden ons bezig met cloud transities met hoofdzakelijk Azure. We zetten actief in op de professionele en persoonlijke ontwikkeling van onze collega's, waarbij persoonlijke aandacht voorop staat.

- Fijne en ongedwongen cultuur
- Hybride werken
- Flexibele werktijden
- Bovengemiddelde arbeidsvoorwaarden
- Onbeperkt mogelijkheid tot ontwikkeling
- Hoog kennisniveau en senioriteit
- Sociale teamevenementen
- En nog veel meer

Scan de QR code voor meer informatie en solliciteer direct

DAPR

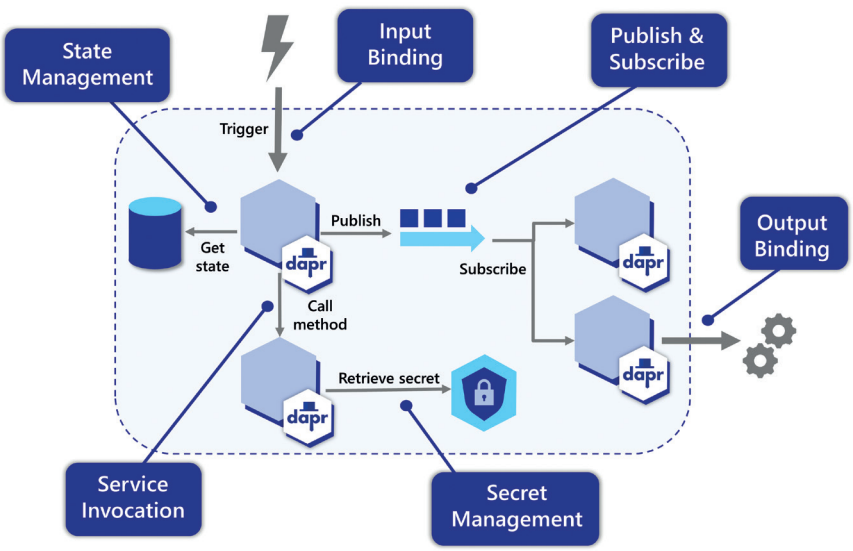
Dapr ofwel Distributed Application Runtime biedt een framework waarbij de complexiteit van Service Discovery, Service Bus integratie, secrets management, observability en encryptie uit handen genomen wordt, zodat de developer zich kan focussen op het schrijven van business logica.

Auteur: Stefan van Tilborg

Er bestaan meerdere tools en frameworks om gedistribueerde applicaties te maken zoals Orleans, Istio en Akka. Dapr onderscheidt zich van dit rijtje door het gebruik en beschikbaar stellen van "building blocks" waarmee je een gedistribueerde applicatie kunt opbouwen: Service Invocation pub / sub state management secret management input / output bindings virtual actors Dapr is daarbij wel afhankelijk van een container platform zoals Kubernetes om op te kunnen draaien.

Sidecar

Al deze building blocks zijn eenvoudig in een applicatie te integreren en nemen het werk van taken als service discovery, state en connectiviteit voor je uit handen. Dapr doet dit door met elke service een sidecar container mee te deployen. Een sidecar container draait naast de service container en voegt functionaliteit toe aan deze container, zonder deze daadwerkelijk aan te passen. De service maakt verbinding met de API's binnen de sidecar en deze regelt het vinden en verbinden met andere services.
De buiding blocks zijn beschikbaar voor een grote keus aan platformen. Wanneer de app gedeployed wordt op Azure dan kan pub/sub worden geconfigureerd voor Azure Servicebus, wordt er gedeployed naar AWS dan kan die zelfde pub/sub gebruikmaken van AWS SNS/SQS. De app merkt daar niets van, die communiceert alleen via de sidecar en hoeft niets te weten van de onderliggende infrastructuur.



CLI

Voordat je gebruik kan maken van dapr dien je eerst de CLI installeren en de development omgeving initialiseren. Installeer de Dapr cli vanuit: https://docs.dapr.io/getting-started/install-dapr-cli/. Na installatie kan deze middels het volgende commando geinitialiseerd worden: **Voorbeeld 1**.
De CLI deployed nu 3 containers, Redis voor onder andere state en pub/sub, zipkin voor tracing en dapr_placement voor de sidecar functies. Om een applicatie te starten met Dapr kun je lokaal gebruik maken van de Dapr CLI, deze koppelt de sidecar aan je applicatie: **Voorbeeld 2**.
Bovenstaand commando zal in de project folder van je service het dotnet project starten. Het app-id wordt door de sidecar gebruikt als identificatie van de service en de app-port is de port waarop de sidecar met de service kan verbinden.

Zeeslag

Bijna elke demo applicatie valt terug op dezelfde shopping cart implementatie. Om het eens helemaal over een andere boeg te gooien, gaan we in dit artikel een gedistribueerde versie van Zeeslag maken. Schematisch ziet deze er als volgt uit: De code is te vinden op: https://dev.azure.com/XPRTZ/_git/Dapr_Zeeslag. De zeeslag applicatie bestaat uit 4 backend services en een blazor frontend (client en Backend For Frontend) voor de user interface. De Player service is een API die de spelers beheert en wordt aangeroepen met service invocations. De Game service beheert het lopende spel en de board service het huidige bord met schepen, waarbij beide services zijn aan

```
dapr init
```
1

```
dapr run --app-id myapp --app-port 5000 -- dotnet run
```
2

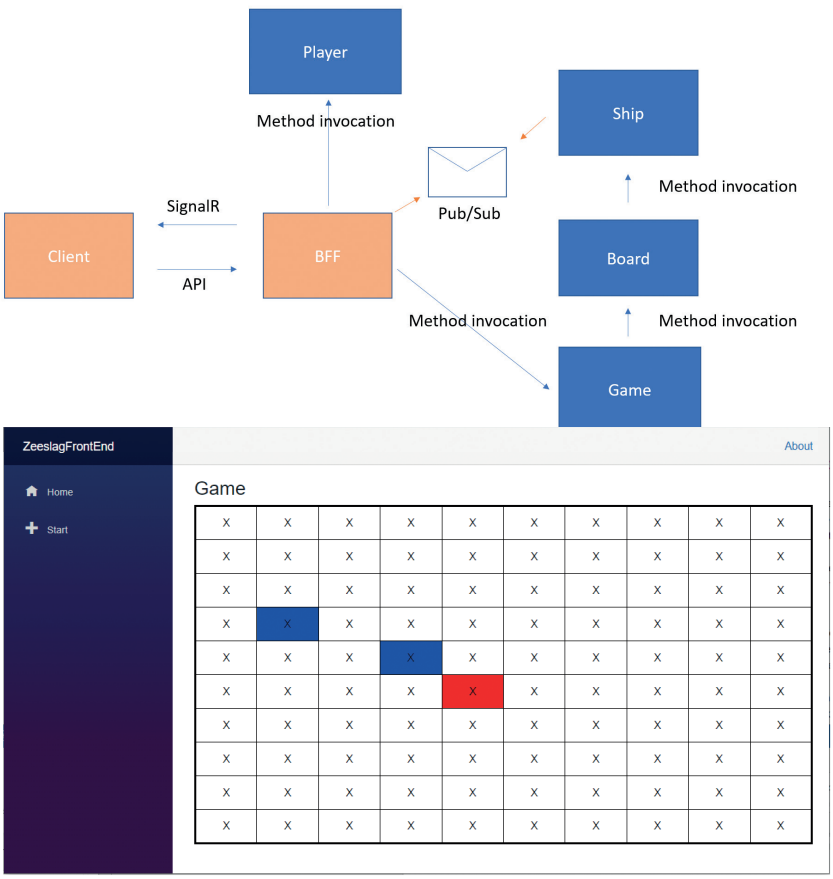
```
DaprClient client = new DaprClientBuilder().Build();
```
3

```
var ship = new Ship(new Guid(), boardId, 5,
    new Point(4, 2), new Point(4, 7));

var shipRequest = client.CreateInvokeMethodRequest(HttpMethod.Post,
    "shipservice", "ships", ship);
```
4

```
var shipResult = await client.InvokeMethodAsync<Ship>(shipRequest);
```
5

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



te roepen met service invocations. De Ship service beheert de schepen die geplaatst zijn op een bord en communiceert met de overige services met pub/sub voor de uitgevoerde schoten en opgeblazen schepen. De implementatie van de door de zeeslag applicatie gebruikte building blocks, Service Invocations, Publish & Subscribe en state management worden in de volgende paragaven beschreven.

Service Invocations

Service invocations is de request / response implementatie van Dapr. Je hebt hierbij geen URL locatie nodig van de service die je aan wil roepen, alleen het app-id. Door gebruik te maken van de Dapr.Client nuget package kan de DaprClient gebruikt worden in ons project. Deze wrapt de http communicatie en biedt een aantal methoden om requests te maken en responses uit te lezen. Dapr kan ook gebruikt worden op basis van http of grpc zonder dat de API aangepast hoeft te worden, op basis van conventie. In de verdere examples is gebruik gemaakt van de DaprClient. **Voorbeeld 3**.
Met de client kan een request gemaakt worden op basis van HTTP method, app-id van de aan te roepen service, de methode van de aan te roepen service en optioneel het te serialiseren object als body: **Voorbeeld 4**.
De client wordt daarna gebruikt om het request te versturen: **Voorbeeld 5**.

Publish & Subscribe

Om gebruik te maken van pub/sub moet eerst de WebApplication geconfigureerd worden door MapSubscribeHandler aan te roepen. Bij het starten van de service met de sidecar wordt op dat moment alle met het Topic attribute geannoteerde controller operaties of, bij minimal API's (zoals in het voorbeeld), de route handlers geregistreerd bij de Dapr runtime. **Voorbeeld 6**.
Vanaf dat moment kunnen de gepubliceerde berichten gerouteerd worden naar de handlers. Onderstaand is een snippet code hoe de BFF een bericht publiceert van een schot op het bord: **Voorbeeld 7**.

State Management

De zeeslag applicatie maakt gebruik van state management om de plaatsing van schepen en geloste schoten

```
var app = builder.Build();

app.MapSubscribeHandler();

app.MapPost("/shots", [Topic("pubsub", "shots")] async (Shot shot) => { … }
```
6

```
[HttpPost]
[Route("shoot")]
public Task Shoot([FromBody] Shot shot)
{
    return _daprClient.PublishEventAsync("pubsub", "shots", shot);
}
```
7

```
const string DAPR_STORE_NAME = "statestore";
```
8

```
await client.SaveStateAsync<Board>(DAPR_STORE_NAME,
        $"BS_{newBoard.Id}", newBoard);
```
9

```
var board = await client.GetStateAsync<Board>
        (DAPR_STORE_NAME, $"BS_{boardId.ToString()}");
```
10

```
await client.DeleteStateAsync(DAPR_STORE_NAME,
            $"BS_{boardId.ToString()}");
```
11

bij te houden voor een gestarte game. Om gebruik te maken van state management moet een state provider geconfigureerd worden in de statestore.yaml in de .dapr/components folder. Out-of-the-box krijg je een Redis statestore die al is voor-geconfigureerd door het dapr init commando. De volgende statestore naam moet opgegeven worden bij gebruik van de Dapr-Client: **Voorbeeld 8**.
Opslaan van een te serializeren object gaat dan als volgt met een string key: **Voorbeeld 9**.
Ophalen en de-serialiseren kan dan met de GetStateAsync methode met gebruik van de string key: **Voorbeeld 10**.
Verwijderen van state gebeurt met de methode DeleteStateAsync: **Voorbeeld 11**.
Dapr bied ook een mogelijkheid om te querieën op de state data, dit bleek in de praktijk niet soepel te werken, in de zeeslag applicatie is hier omheen gewerkt door de keys te prefixen.
Behalve Redis zijn een groot aantal state providers beschikbaar zoals CosmosDB en verschillende SQL Databases.

Debugging

Een Dapr applicatie "out of the box" debuggen wordt voor dotnet alleen ondersteund in VS Code. Door in de tasks.json de Dapr parameters als app-id en app-port te configureren, en in de launch.json in de parameter preLaunchTask te verwijzen naar de Dapr configuratie wordt de applicatie in debug mode gestart en kan door de applicatie code heen gestept worden. Als in de launch.json ook de compounds configuratie wordt opgenomen kunnen meerdere services in debug mode gestart worden, wat zeker tijdens development heel handig is.
Een kleine kanttekening; minimaal 1 van de geconfigureerde Dapr services moet als grpc-port 50001, en als http-port 3500 geconfigureerd hebben. Dit is verder niet gedocumenteerd op het moment van dit schrijven, maar wanneer hier niet aan voldaan wordt, start de applicatie niet goed op om dat er geen verbinding met de sidecar gemaakt kan worden.

Final thoughts

Dapr biedt zeker meerwaarde bij het maken van een gedistribueerde applicatie. Het uithanden nemen van boilerplate code voor service to service communicatie en state management werkt goed en intuïtief. De code draait onder Kubernetes, een veel gebruikt platfoprm, en kan met geen of minimale aanpassing op bestaande services toegepast worden, je hoeft dus niet greenfield te starten of gebruik te kunnen maken van wat Dapr te bieden heeft. De local development en debugging voldoen al zijn een paar schoonheidsfoutjes nog wel aanwezig.
Wat wel mist is de adoptie van deze tooling. Een zoekopdracht op google levert maar weinig hits op van bedrijven / instellingen die Dapr daadwerkelijk in productie gebruiken. Dapr is al beschikbaar sinds 2019 en is actief in ontwikkeling, de massale adoptie lijkt wat achter te blijven ondanks dat het zoveel te bieden heeft.
Is Dapr Het go-to platform voor de dotnet developer? Als de adoptie groter zou zijn en de schoonheid foutjes opgelost zijn zou ik zeker volmondig Ja zeggen. Ik heb in ieder geval fijn met het framework gewerkt en ik zou het iedereen aanraden om het zelf eens uit te proberen. ●

BIO

Stefan van Tilborg

Stefan van Tilborg werkt als Dotnet Developer en Solution Architect bij XPRTZ; Dé plek voor .NET experts! Momenteel is hij ingezet bij Rademaker, als Solution Architect Digitalisering. Hij heeft een passie voor software development en integratie vraagstukken. Buiten werktijd mag hij graag zijn Lego automatiseren met Mindstorms, iot-dotnet en BrickPI.

Architectuur design **met het C4-model**

Auteur: Jacob Duijzer

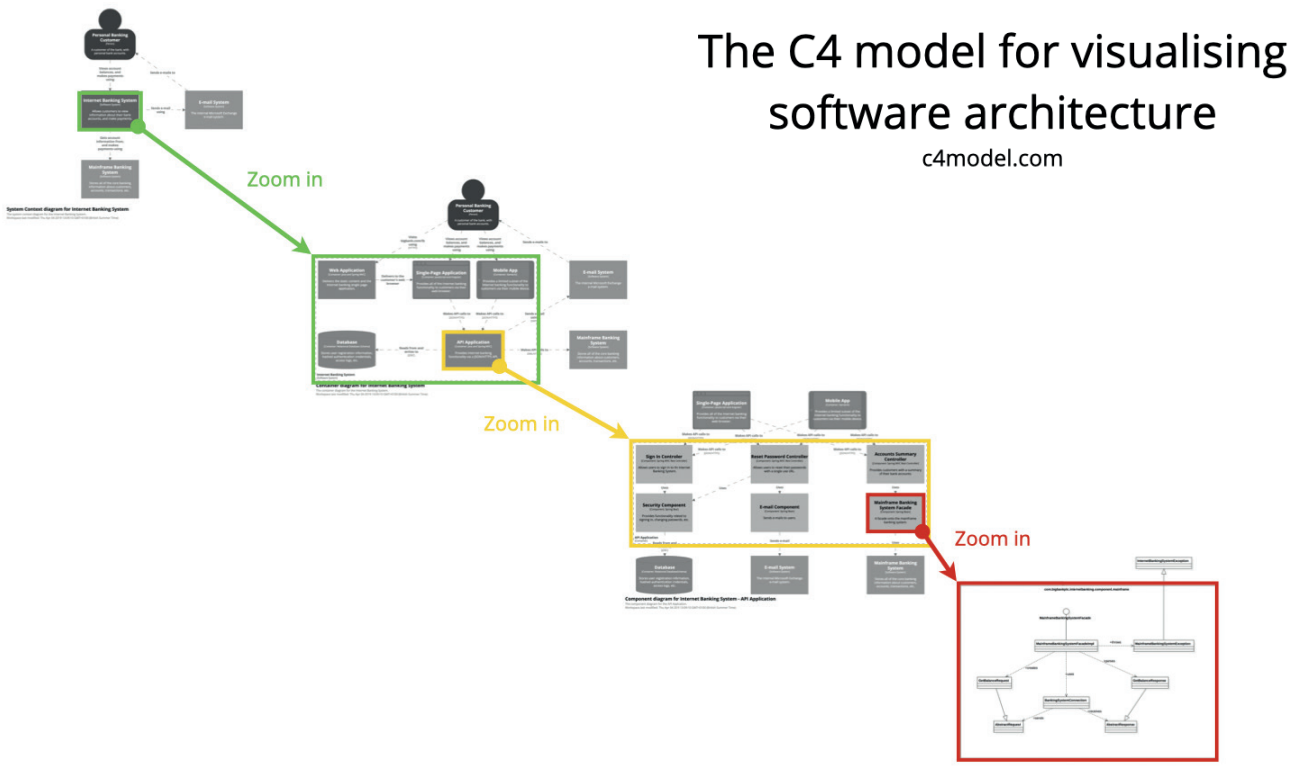
Samenwerken. Dialoog. Conversatie. Elk boek over softwareontwikkeling besteed er aandacht aan. Het is *misschien wel* het belangrijkste middel om succesvol software te ontwikkelen. Of het nu gaat over domeinen, specificaties of architectuur, uiteindelijk moet er een ondubbelzinnig beeld ontstaan van het te ontwikkelen Software Systeem. Ook bij het ontwerpen van de architectuur is de dialoog belangrijk. Het is daarbij van belang dat alle betrokkenen begrijpen hoe de architectuur van het softwaresysteem er uit ziet.

Verschillend publiek

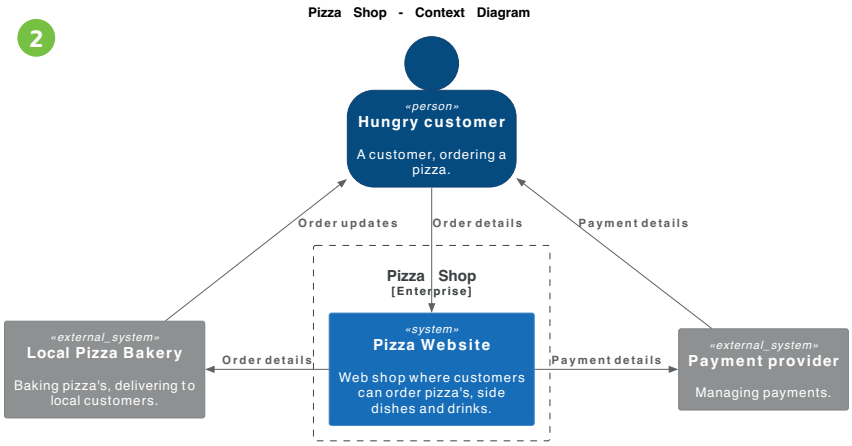
Bij het ontwikkelen van software zijn veel verschillende partijen betrokken. Stakeholders, Product Owners, System Engineers, Testers, Software Engineers, elke discipline heeft een ander belang bij de architectuur. Een stakeholder wil dat er waarde gecreëerd wordt, dat er een probleem wordt opgelost, een system engineer zal willen weten hoe het beheer moet gebeuren. Bij het ontwerpen van een software systeem moeten we het gesprek dus aanpassen aan het publiek. Elke weergave moet echter wel een waarheid weergeven om te voorkomen dat er designs ontstaan die in de praktijk niet gebruikt worden.
Het C4-model kan helpen om de architectuur van een software systeem te visualiseren door het systeem in verschillende lagen te ontleden, elk niveau met zijn eigen detaillering.

Het C4-model

Context, Containers, Componenten en Code. Dat zijn de 4 C's van het C4-model. Het heeft niets met het explosief C4 te maken, al kan het



1



toepassen van C4 wel een explosieve impact hebben. Het is ontwikkeld door Simon Brown, die bij workshops tot de conclusie kwam dat visualiseren van architectuur een vaardigheid is die bij veel developers en architecten ontbreekt.
Figuur 1: Het C4-model.

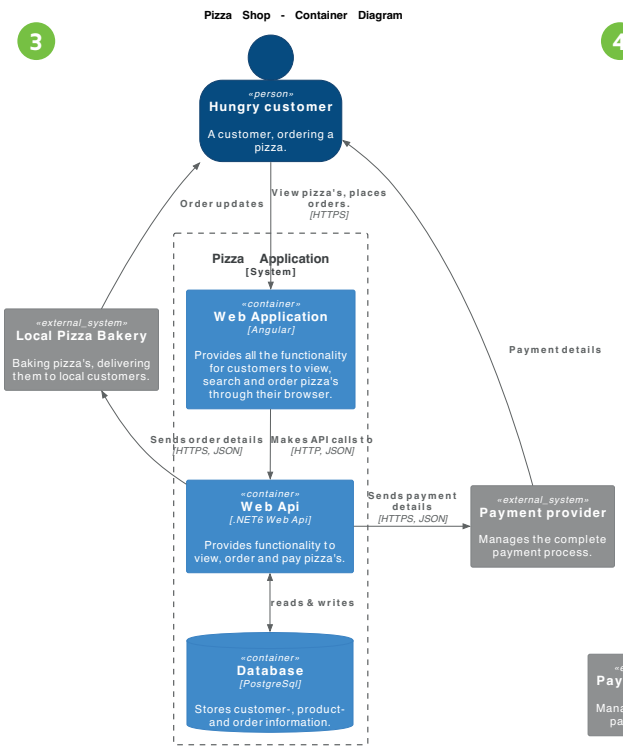
Het C4-model beschrijft de architectuur van software systemen door het te ontleden in verschillende niveaus, van de context tot de code. Het model beschrijft diverse elementen die kunnen helpen om gezamenlijk een ontwerp te maken en te visualiseren. Door de niveaus gescheiden te houden blijft de complexiteit van een design lager en is het uitermate geschikt om in een snel veranderende wereld gebruikt te worden.

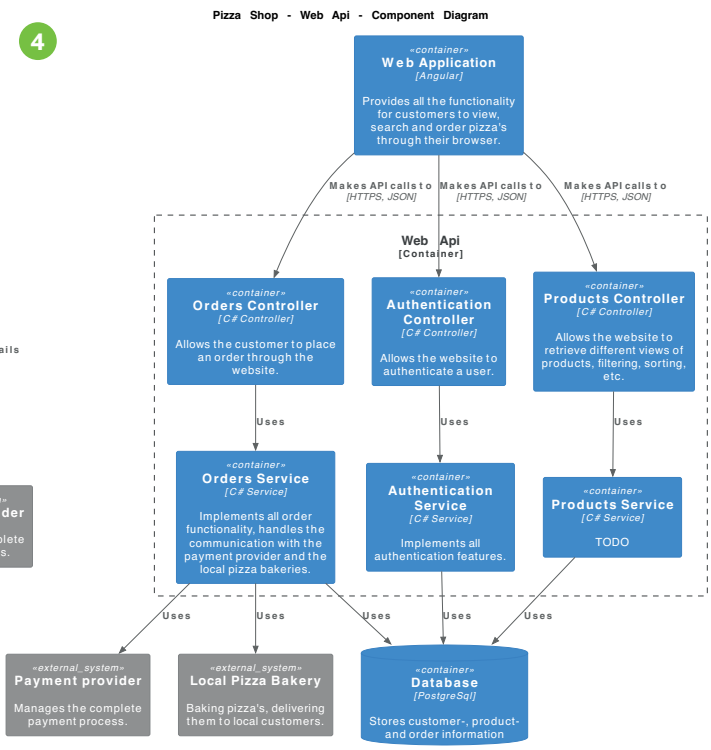
Context

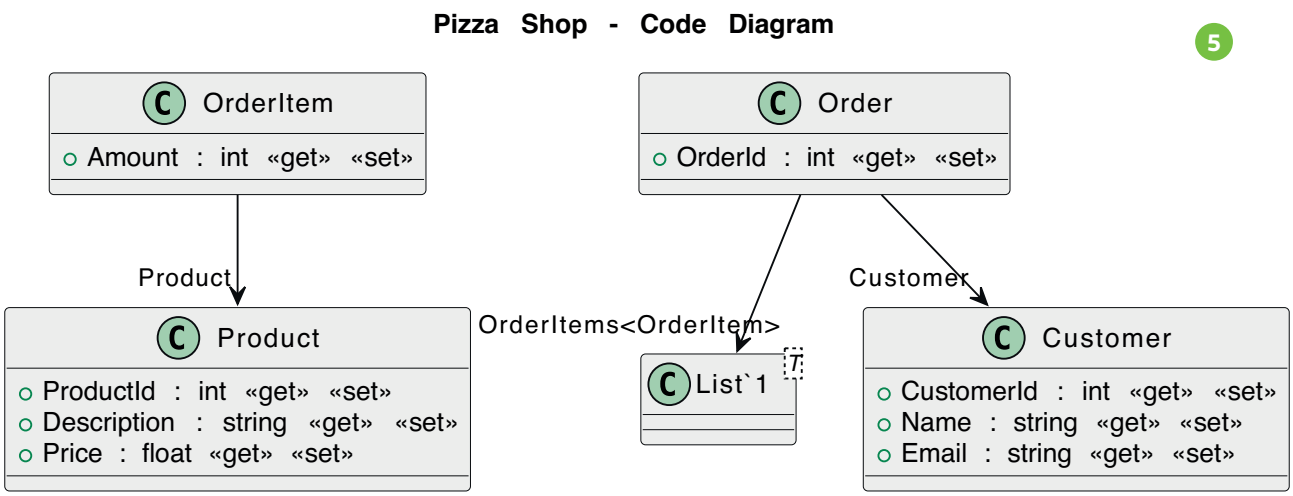
Het context diagram is vaak een goed punt om mee te beginnen. Het geeft een goed beeld van het hele landschap, het geeft het eigenlijke software systeem weer met de belangrijkste gebruikers en externe systemen waar het systeem van afhankelijk is. Details zijn hierbij niet belangrijk, die komen in diepere lagen aan bod.
Figuur 2: een context diagram.
In dit artikel gebruik ik het voorbeeld van een website waar je pizza's kunt bestellen. Pizza's bestel je op een website, je betaald je bestelling en de lokale pizzabezorger komt de pizza thuis bezorgen. Zowel het betaalsysteem als het systeem van de lokale pizzabezorger zijn buiten de scope van ons softwaresysteem maar wel onderdeel van het grote geheel.

Containers

Elk softwaresysteem is opgebouwd uit verschillende containers. Het gaat hier niet om Docker containers, C4 gebruikte deze term al voordat Docker populair werd. Denk bij een container aan een webapplicatie, een API of een database. In dit diagram gaat het er vooral om







welke containers er zijn en wat de onderliggende relaties zijn.
Figuur 3: een container diagram.
Er begint nu meer duidelijk te worden over de architectuur van het systeem. Blijkbaar bestaat het systeem uit een Angular webapplicatie, een Web API en een database. De API lijkt het meest complex, ook vanwege de communicatie met externe systemen, maar er zijn nog te weinig details beschikbaar.

Component

Het component diagram geeft een detailweergave van een container. Welke belangrijke onderdelen zijn er nodig om een container te ontwikkelen, wat zijn de taken van elk onderdeel en wat zijn belangrijke technische implementatiedetails.
Figuur 4: een component diagram.

Code

Vreemd genoeg wordt het gebruik van het vierde niveau, code, afgeraden. Althans, zo vreemd is het niet. De meeste software-ontwikkelprogramma's zijn, al dan niet met behulp van plug-ins, prima in staat om UML-diagrammen te genereren op basis van code. Zo heb ik hieronder een voorbeeld van UML, vanuit code gegenereerd. Code wijzigt vaak sneller dan de architectuur en het handmatig bijhouden van code diagrammen kan dan een tijdrovende en overbodige taak worden.
Figuur 5: een UML diagram voor code.

Het bovenstaande voorbeeld komt bijvoorbeeld uit de JetBrains Rider IDE, het genereren van dergelijke diagrammen is goed te automatiseren, zelfs het automatisch updaten van documentatie is daardoor mogelijk.

Tools

C4 diagrammen kunnen met verschillende tools gemaakt worden. Er zijn plug-in's beschikbaar voor de meeste software-ontwikkelprogramma's, er is een online editor beschikbaar, het kan zelfs met websites als draw.io. Denk er bij het ontwerpen wel aan dat versiebeheer onmisbaar is. Online websites maken het werken met C4-modellen een stuk lastiger. Ontwerpen en aanpassen vanuit een IDE of andere editor maakt het werk vaak een stuk eenvoudiger.
Figuur 6: Een C4-diagram in de JetBrains Rider IDE.

Het bovenstaande voorbeeld maakt gebruik van de PlantUml imple-

```
@startuml

!include https://raw.githubusercontent.com/plantuml-stdlib/C4-PlantUML/master/C4_
Context.puml

title Pizza Shop - Context Diagram

Person(client, "Hungry customer", "A customer, ordering a pizza.")

Enterprise_Boundary(pizza_enterprise, "Pizza Shop") {
    System(pizza_app, "Pizza Website", "Web shop where customers can order pizza's,
side dishes and drinks.")
}

Rel(client, pizza_app, "Order details")

System_Ext(payment_provider, "Payment provider", "Managing payments.")
System_Ext(local_pizza_shop, "Local Pizza Bakery", "Baking pizza's, delivering to
local customers.")

Rel_R(pizza_app, payment_provider, "Payment details")
Rel_L(pizza_app, local_pizza_shop, "Order details")

Rel_U(payment_provider, client, "Payment details")
Rel_U(local_pizza_shop, client, "Order updates")
@enduml
```

7

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

mentatie van het C4-model. Deze implementatie is beschikbaar via GitHub. Het contextdiagram, het eerste diagram in dit artikel, ziet er in code zo uit: **Figuur 7**

Het is mogelijk om de stijl aan te pakken, kleuren te veranderen, eigenlijk is alles aan te passen. Let er op dat je vooral aandacht besteed aan de duidelijkheid van de diagrammen, kleuren voegen daar lang niet altijd iets aan toe!

Conclusie

Eén van de weinige keren dat ik software documentatie daadwerkelijk gebruikt heb zien worden was bij projecten waar het C4-model[1] gebruikt was. De hiërarchische aanpak helpt om de juiste detaillering te geven. Opdrachtgevers en architecten gebruiken de context en containers, Software Engineers kijken sneller naar de details van de componenten en de samenhang hiervan bij de containers, Systeem Engineers helpt het om inzicht te krijgen in beheersvraagstukken.

De eenvoud en eenduidigheid maakt dat het C4-Model snel geadopteerd wordt en kennis van het C4-model[1] is bruikbaar bij alle projecten. Door de verschillende niveaus op te splitsen blijft elk niveau relatief eenvoudig en daarmee goed onderhoudbaar, uitermate geschikt van voor Agile projecten. Als een bestaand project slecht gedocumenteerd kan het teams helpen door interactief aan de slag te gaan met het C4-model[1]. Ontwerp gezamenlijk de verschillende diagrammen waarbij door middel van discussies veel kennis opgedaan kan worden van het reeds bestaande systeem.

Wil je meer weten van het C4-model[1]? Bekijk dan vooral de presentatie Visualising software Architecture with the C4 model van Simon Brown2 op de C4-model website. ●

BIO

Jacob Duijzer

Jacob Duijzer is een Software Engineer met een passie voor kwaliteit en samenwerking. Hij is altijd op zoek naar de verbinding tussen diverse disciplines en het verbeteren van teams, vaak door middel van het organiseren van Coding Dojo's. Jacob is werkzaam bij Team Rockstars IT.

LinkedIn: https://www.linkedin.com/in/jacobduijzer/

Bronnen

Het C4-model: https://c4model.com
Simon Brown: https://simonbrown.je
Visualising Software Architecture with the C4-model: https://youtu.be/x2-rSnhpw0g
PlantUml: https://plantuml.com
C4-PlantUML: https://github.com/plantuml-stdlib/C4-PlantUML

Referentie

[1]: https://youtu.be/x2-rSnhpw0g

Intervisie in de praktijk van IT- consultants

Auteurs: Erik van Olst en Arjen Kraak

In de gezondheidszorg is intervisie gemeengoed. We hebben ons verdiept in de toepasbaarheid van intervisie als methodiek in de wereld van IT. Aan de hand van enkele vragen schetsen we een IT-perspectief van de intervisie methodiek.

Wat is intervisie?

Intervisie is een gesprekstechniek waarbij deelnemers door middel van vragen en discussie een andere deelnemer helpen onder begeleiding van een intervisie begeleider.

Welke rollen zijn er nodig?

Voor een intervisie sessie is iemand nodig die een casus inbrengt.
De casus inbrenger is de persoon die de hulpvraag heeft.

De intervisie sessie wordt begeleid door de intervisie begeleider.
Deze persoon staat los van de inhoud en stuurt strak op het proces. Hierbij bewaakt de intervisie begeleider de veiligheid en vertrouwelijkheid. De intervisie begeleider stelt zelf geen inhoudelijke vragen en stimuleert de deelnemers deze vragen te stellen.
De deelnemers zijn de personen die de casus inbrenger willen helpen zodat de casus inbrenger zelf tot nieuwe inzichten komt.

Wat is intervisie niet?

Intervisie is niet gericht op teams. Het is ook geen methodiek voor een 1 op 1 coaching tussen twee personen. Het is niet bedoeld om functioneren te beoordelen en ook niet voor feedback of (team)verbetering.

Wanneer pas je het toe?

Als de casus inbrenger vastloopt in een bepaalde situatie, of geen zicht heeft op zijn/haar gedrag, dan kan intervisie de methodiek zijn om dit gedrag beter te begrijpen en actie(s) te bepalen.

> Intervisie is niet gericht op teams en niet bedoeld om functioneren te beoordelen

Binnen de wereld van IT, en softwareontwikkeling in het bijzonder, zou je intervisie kunnen toepassen op de volgende situaties:

> Ontwikkelaar 'Peter' (fictieve naam) heeft moeite zich uit te spreken tijdens een standup. Hij voelt zich ongemakkelijk om in het publiek te spreken over zijn werk
> De Scrum Master van Team X heeft het dilemma dat het team geen tijd wil maken voor het refinen van toekomstig werk

Wat levert het op?

Na afloop van een intervisie sessie heeft de casus inbrenger inzicht gekregen in zijn/haar wenselijk gedrag en/of acties die noodzakelijk zijn om de casus uitdaging aan te gaan. Als er vaker een intervisie wordt gehouden met dezelfde deelnemers ontstaat betere verbinding en vertrouwen naar elkaar.

Wat zijn de huisregels voor een intervisie sessie?

Voor een waardevolle intervisie sessie zijn een aantal huisregels van groot belang waar de intervisie begeleider op moet toezien. Deze zijn:

> De intervisie begeleider moet ervoor zorgen dat er binnen de deelnemers bevestiging komt over vertrouwelijkheid en veiligheid. Wat binnen de kamers van de intervisie sessie wordt besproken, blijft binnenskamers
> Er moet een volledige mate van gelijkwaardigheid zijn tussen de deelnemers. Met andere woorden, hiërarchie mag niet aanwezig zijn tijdens de sessie. Mocht hiërarchie een rol spelen, komt de vorige huisregel over veiligheid en vertrouwelijkheid in gevaar
> De deelnemers moeten focus hebben tijdens de intervisie sessie en beschikbaar zijn
> De deelnemers mogen geen gezamenlijk belang hebben over de casus die wordt ingebracht. Een gezamenlijk belang geeft spanningen voor de eerste huisregel

Hoe gaat het in zijn werk?

De intervisie sessie wordt doorlopen in een drietal stappen:

> Bevestigen van de setting (veiligheid, vertrouwelijkheid, focus en beschikbaarheid)
> De intervisie sessie zelf
> Bevestiging actiepunten (indien van toepassing)

De intervisie sessie zelf wordt uitgevoerd in een aantal onderdelen:

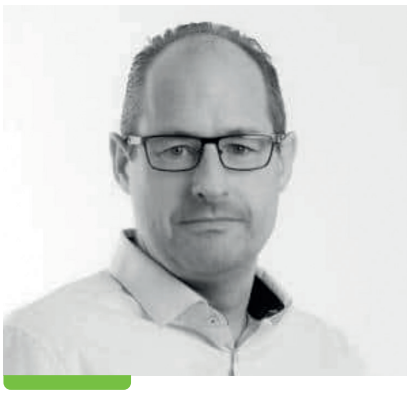
> Keuze van de casus
> Verdieping van de casus; door het stellen van ten eerste verhelderende en vervolgens verdiepende open vragen aan de casus inbrenger
> Deelnemers discussiëren zonder inmenging van de casus inbrenger de situatie en bespreken mogelijkheden voor gewenst gedrag. De casus inbrenger luistert en notuleert
> De casus inbrenger reflecteert wat hij/zij heeft gehoord en geleerd. De casus inbrenger vertelt wat zijn/haar actie(s) is om zichzelf verder te helpen
> De deelnemers delen wat zij zelf hebben geleerd van de casus
> Het gevolgde proces wordt geëvalueerd

Welke valkuilen hebben we ontdekt?

Als IT-consultant hebben we vaak een leidende rol, daarbij gesteund door onze domeinkennis.
Uit de praktijk blijkt dat we als begeleider moeten voorkomen inhoudelijk te sturen en daarentegen met veel nieuwsgierigheid de groep moeten observeren. Dit voelt wat onwennig maar de voldoening als de casusinbrenger de sessie met een aantal concrete handvatten uitstapt maakt dat meer dan goed.
Een tweede uitdaging: pas géén intervisie toe op teams die al samenwerken. Zo hebben we binnen ons eigen team eens een casus behandeld die tijdens de sessie een gezamenlijk dilemma bleek te zijn. Het feest der herkenning werd gewaardeerd, maar het gezamenlijk belang zorgde ervoor dat vervolgacties uitbleven. Een brainstormsessie of visgraatanalyse was hier waarschijnlijk meer op zijn plaats geweest.
Al doende hebben we zo gemerkt dat de intervisieregels er niet voor niets zijn en moeten kiezen voor een alternatief als we niet aan alle voorwaarden kunnen voldoen.

We sluiten af met de hoop dat we intervisie steeds meer terug gaan zien in onze sector. In de zorg is vele jaren geleden al bewezen dat het in de praktijk écht werkt.

De intervisie begeleiders van Bergler Software Solutions,
Erik van Olst en Arjen Kraak ●

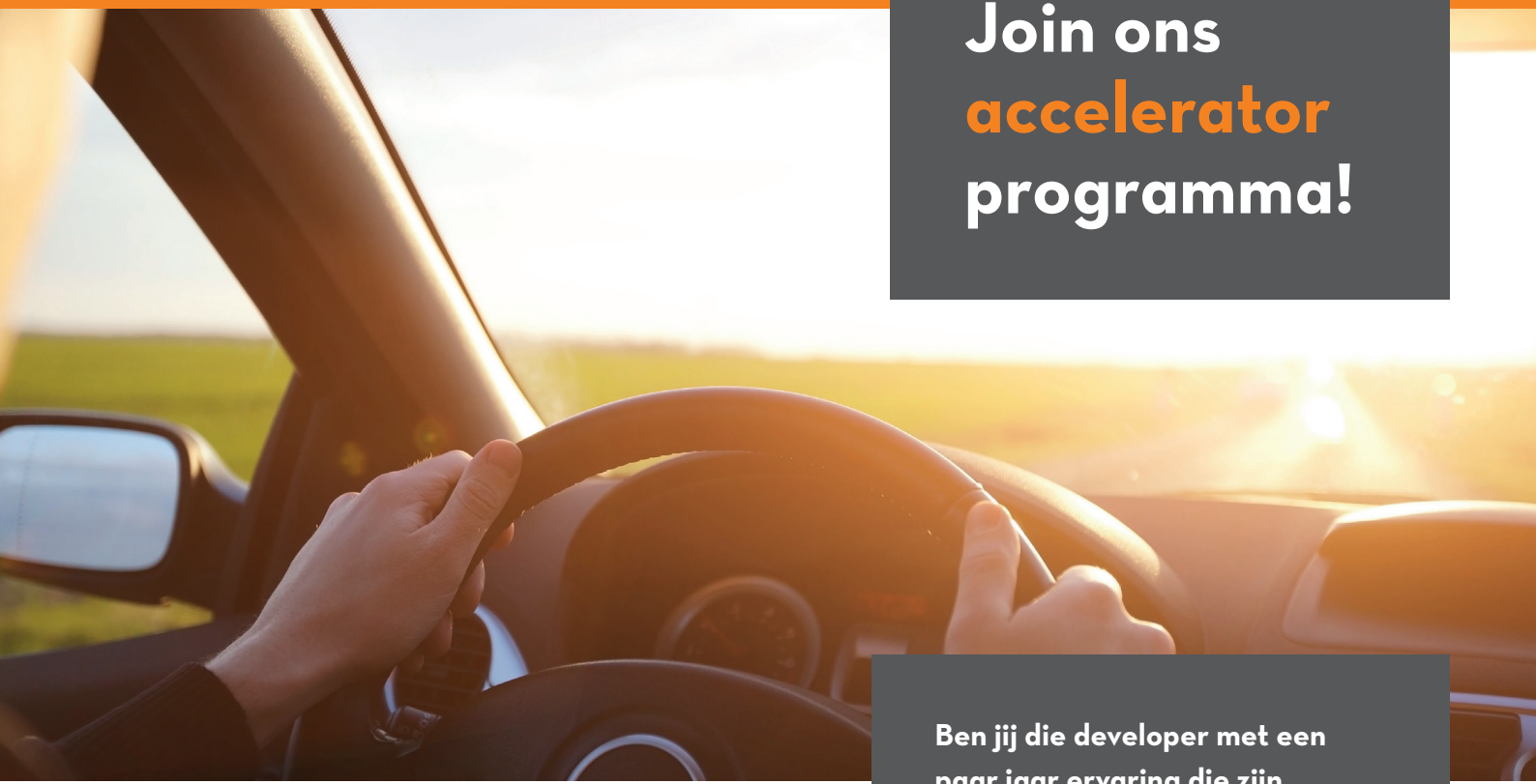


Arjen Kraak,Technical Lead Consultant bij Bergler Software Solutions

Erik van Olst, Technical Lead Consultant bij Bergler Software Solutions

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

Join ons accelerator programma!



ORDINA

Software Development

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

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

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

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

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