# Dia 1: Titel pagina

Welkom iedereen bij de presentatie van mijn bachelorproef. Mijn bachelorproef ging over microservice integration patterns op SAP order-to-cash process. Dit onderwerp werd voorgesteld door delaware. Ik wil ook Nicolas Pauwelyn bedanken om mijn co-promotor te zijn.

Ik ben Lyva Van Damme, studente toegepaste informatica.

# Dia 2: inhoudstabel

Vandaag ga ik volgende onderwerpen aanhalen:

* Stand van zaken
  + Microservices
  + Order-to-cash proces
  + Requirements van de business
* Methodologie
  + Enkele termen
  + De microservices binnen het order-to-cash process
  + Databank structuur
  + Complete architectuur
* Conclusie

# Dia 3: Inleiding

Tussentitel dia

# Dia 4: Waarom microservices en het order-to-cash process?

Microservice is een opkomende technologie die invloed heeft op de architectuur van applicaties en processen. Er wordt naar gerefereerd als de tegenhanger van monolithic. De termen worden later uitgelegd.

Er werd gekozen voor het order-to-cash proces omdat dit proces verschillende afdelingen binnen een bedrijf overlapt.

# Dia 5: Stand van zaken

Tussentitel dia

# Dia 6: Microservices

Hier bespreek ik:

* Definitie
* Belang van microservices
* Algemene aanpak om microservices te implementeren
* Voordelen en nadelen van microservices

# Microservices: definitie

In deze dia ga ik microservices uitleggen.

Een term die vaker zal voorkomen is monolithic. Monolithic is de “tegenhanger” van microservices. Uitleggen aan de hand van de figuur.

Microservices, 3 onderdelen:

* Onafhankelijk, kleine, modulaire services: Modulaire services zijn services waarbij veel delen uitwisselbaar zijn met diverse services. Wordt er info gestuurd of gevraagd van services A dan zal dit geen invloed hebben op de andere services.
* Eenvoudige communicatie: Er is nood aan omdat sommige services data moeten uitwisselen om hun ’job’ te kunnen doen. De communicatie kan gebeuren op verschillende manieren. De manier die gekender is, is ’Messaging via Message Broker’. Dit wil zeggen dat microservice A een bericht plaatst op de wachtrij bij microservice B wanneer die data wil doorsturen. Dan kan microservices B aan die data wanneer hij die nodig heeft. Ze zullen soms moeten wachten maar ze zijn zo goed als onafhankelijk van elkaar.
* Gemaakt in functie van business requirements

# Belang van microservices

Redenen:

* Het is gemakkelijker om kleine services te onderhouden.
* Microservice onafhankelijk deployen
* Gemakkelijker om aan te passen aan nieuwe technologie
* Eenvoudgier om te schalen
* No single point of failure
* Oplevering kan sneller gebeuren

## De verschillende manieren van bescherming

De manieren van bescherming:

* Coderingsstandaarden
* Toegangscontrole
* Documentatie
* Garantie op data privacy
* Encrypteren van data
* Oauth voor identificatie gebruikers
* Firewall
* Monitor
* Throtteling: is a process used to control the usage of APIs by consumers during a given period.

## Authenticatie en authorisatie

Aan de hand van gebruikersnaam en wachtwoord.

Een API gateway uitleggen.

## Verband met Agile en Devops

Ideologie: afbreken van kleine, traag evoluerende architectuur of monolithic en deze in microservices steken.

Nadruk op samenwerken en communicatie tussen verschillende partijen.

DevOps: a methodology that enables developers and IT Ops to work closer together so they can deliver better quality software faster.

Agile: software in kleinere delen opsplitsen om op kortere periodes kleinere deeltjes software op te leveren.

Voordelen van de combinatie microservices en DevOps:

* Meer opleveringen van software op kortere periodes
* Betere kwaliteit van de code
* Software herbruiken
* Hoger level van automatisatie

## Het monitoren van microservices

Metrics zijn numerieke waarden die kunnen geanalyseerd worden. Logs komen voor op volgende niveau’s van een applicatie.

Het monitoren of loggen van microservices houdt in dat er wordt bijgehouden hoe een microservice zich gedraagt.

Elke microservice loggen, om later de oorzaak van een fout terugvinden.

Best practices:

* Niet in bestanden opslaan. Streams van flows.
* Logging moet werken over alle verschillende codeertalen.
* Elke requist van logging uniek ID.
* Structuur in de log data.

# Algemene aanpak om microservices te implementeren

Voor het overschakelen:

* Hoe zijn andere bedrijven overgeschakeld?
* Ervaringen van anderen lezen
* Plan opmaken

Het 6-stappen plan:

1. Serve a business purpose  
   Business requirement omzetten naar een microservice  
   Bijvoorbeeld ophalen van data om die dan te analyseren en conclusies uit te trekken.
2. Protect your stuff  
   Bescherming van de microservice.  
   Uniforme manier van bescherming toepassen.  
   Bescherming is er voor zorgen dat de user enkel kan wat hij hoort te kunnen en zich ook enkel kan aanmelden als hij het echt is.
3. See no evil, hear no evil  
   Monitoren van de microservices.  
   Het gedrag van de microservices bijhouden en kunnen analyseren bij problemen.
4. Find your stuff  
   Hoe de microservices met elkaar kunnen communiceren.  
   Service registry: waar de microservices moeten registeren om de data van een andere microservices te ontvangen.
5. Create a gateway  
   API gateway: scherm waar je gegevens op invult en mogelijke acties op doet en die spreken dan de correcte microservices aan.  
   Requests en aanvragen naar de juiste microservice worden doorgestuurd.
6. Construct events  
   Asynchrone verwerking.  
   De berichten kunnen op eigen tempo verwerkt worden.

# Voordelen en nadelen van microservices

|  |  |
| --- | --- |
| Voordelen | Nadelen |
| Architectuur wordt flexibeler | Bij nieuwe technologieën is het concept niet altijd meteen duidelijk. |
| Hermodeleren, implementeren van nieuwe technologieën eenvoudiger. | Complexiteit |
| De snelheid van de architectuur zou verbeteren | Debuggen en fouten vinden is moeilijker |
| Geen onnodige stappen maken |  |
| Onafhankelijkheid |  |
| Bij falen, valt maar één deeltje uit |  |

# Order-to-cash proces

Definitie aanhalen

Welke technologie biedt SAP aan?

## Definitie

Het proces uitleggen.

## Technologie

**Sap bied Kyma aan**

Open-source project.

Cloud-based en on-premise applicaties.

Betere end-to-end ervarings scenario's.

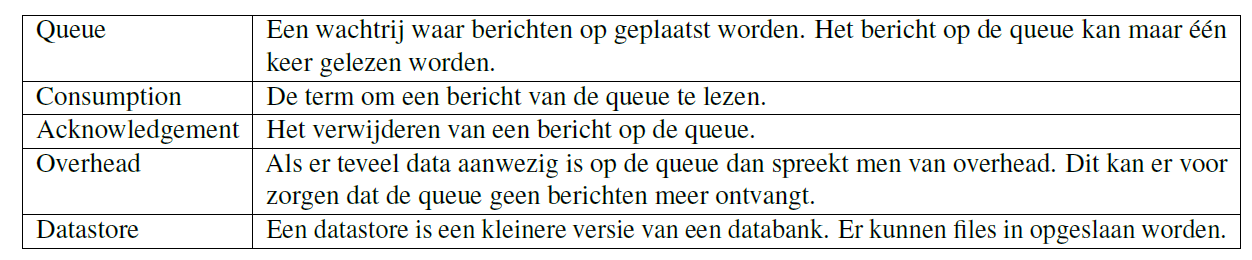
Kleine modules developpen.

**Eigenschappen van Kyma**

* Be open and extendable: Kyma maakt gebruikv an Open Service Broker API specificities. Dit is een “plug and play” die de mogelijkheid geeft om code te hergebruiken. Code van andere partijen te gebruiken.
* Be seamlessly connected: Een eenvoudige manier om een beveiligde connectie te hebben tussen systemen. Deze kunnen gemanaged worden binnen de huidige applicatie om oplossingen te maken in hetrogene landschappen. Eén connectie geeft veel mogelijkheden.
* Use any programming language: Developers kunnen in hun gewenste programmeertaal coderen.
* Bring speed and agility: Er moet niet maanden gewacht worden om use cases of functionaliteiten op te leveren.
* Accelerate innvation: Meestal start dit als een test of een trail. Bij zulke scenario’s zijn de kost en de snelheid van zeer groot belang. Dankzij Kyma kunnen bedrijven meteen beginnen werken aan een oplossing en moet er geen tijd besteed worden aan het zoeken van de best mogelijke oplossing.

Methodologie

# Termen



# Microservices binnen het order-to-cash proces

## De microservices van het order-to-cash process.

Klanten gegevens ophalen

Orders plaatsen, ophalen en verwijderen

Producten ophalen en het aantal in voorraad veranderen

Facturatie maken en ophalen

Shipment documentatie opstellen

Aanmaning opmaken en verwijderen

Berichten plaatsen op de queue

Berichten ophalen van de queue

## De communicatiemethode tussen microservices

De foto uitleggen

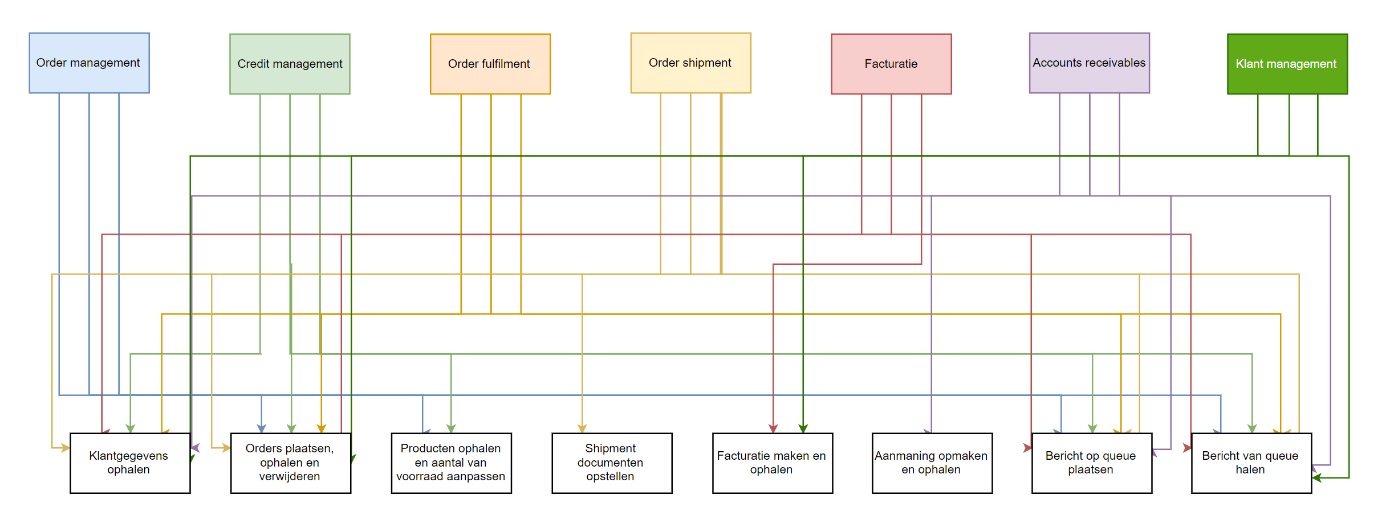
# Databank structuur

De structuur uitleggen en waarom het zo simplistisch is.

# De complete architectuur opbouwen

## De architectuur

De figuur uitleggen.



## De volgende stappen