Software Architecture

# The Bakery (B-MaC)

****

## Probleembeschrijving

Een groot bakkerijconcern genaamd B-MaC vroeg ons om hulp bij het beheren en automatiseren van hun supply chain management en klantgerichte communicatie.

B-MaC levert kwaliteitstaarten, donuts, cakes en koekjes aan andere bakkerijen en detailhandel klanten via een online bestel platform.

Het verschil met de concurrenten van B-MaC is het feit dat je de dag zelf kunt bestellen en dat ze je product dezelfde avond nog kunnen bakken, zodat je je bestelling de dag erna kunt ophalen. Hiervoor hebben ze een magazijn naast de deur waar ze alle ingrediënten opslaan die nodig zijn voor het eindproduct.

B-MaC is erg ambitieus en heeft ons gevraagd ervoor te zorgen dat de architectuur in de loop van de tijd kan evolueren, zodat deze kan meegroeien met de organisatie.

## Gebruikersprofielen

Onze bedrijfsanalisten hadden de gelegenheid om met een aantal belanghebbenden te praten en hebben met succes de volgende gebruikersprofielen vastgelegd:

Het profiel van Dabbing Doughnut.

* **Dabbing Doughnut** is een trouwe klant van B-MaC: We hebben altijd dezelfde bestelling, behalve op zaterdag omdat we op zondag gesloten zijn. We bestellen gewoon elke avond een aantal van elk type donut en halen ze de dag erna op voordat we opengaan. We zijn trouwe klanten en we vinden het leuk dat we per aankoop loyaliteitspunten kunnen verzamelen. We hebben nu een platina klantenkaart en die geeft ons standaard 20% korting, op ons totale order. Dat komt goed van pas als we elke maand donut-donderdag vieren.
  + We hebben een overzicht van eerdere bestellingen, zodat we altijd onze bestelgeschiedenis kunnen controleren en eerdere bestellingen kunnen kopiëren.
  + B-MaC biedt geen gebruikersinterface voor klanten... we integreren rechtstreeks met hun REST API.
* **Barry** is hoofddonut bakker in de bakkerij van B-MaC: We maken de beste donuts in de regio, we hebben meer dan 20 soorten recepten voor de donuts, verschillende soorten, fruit, slagroom enz. We verzamelen de hele dag verkooporders totdat we de cut off-tijd bereiken. Dat is het tijdstip waarop er geen nieuwe bestellingen voor die dag meer worden geaccepteerd en we onze ingrediënten uit het magazijn kunnen bestellen. Tegenwoordig is dit ingesteld op 22.00 uur. We doen dit door een uitgaande bestelling aan te maken in het magazijn. Meestal kan het magazijn de bestelling meteen uitvoeren en krijgen we de ingrediënten meteen geleverd, maar soms krijgen we alleen wat op voorraad is en moeten we wachten op een nieuwe levering. We denken dat ze extra voorraad bestellen bij verschillende leveranciers, maar dat wordt beheerd door Willow, zij is de warehouse manager.
* **Willow**: De uitgaande bestellingen die we krijgen voor ingrediënten zijn heel gevarieerd. In het magazijn hebben we verschillende secties waar we de voorraad kunnen bewaren, fruit moet in onze gekoelde ruimte bewaard worden, suiker en zout in de droge ruimte en sommige ingrediënten hebben een houdbaarheidsdatum, dus die moeten we ook bijhouden.
  + We proberen de bestellingen die we krijgen zo goed mogelijk uit te voeren, maar soms bestellen ze meer dan we op voorraad hebben. Vooral op donderdag kan dit gebeuren. Als onze voorraad op is, moeten we inkomende orders aanmaken bij onze leveranciers. Het zou fijn zijn als we dit niet handmatig hoeven te doen, maar automatisch op basis van het voorraadniveau dat we in het systeem kunnen instellen.

De profiel van de webshop: Fruit Cake Inc.

* **Fruit Cake Inc**. is een nieuwe klant van B-MaC:
  + We zijn op dit moment bezig met het bouwen van onze webshop. We zijn gespecialiseerd in taarten en cakes gecombineerd met fruit: bosbessen, appels, ananas, noem maar op... B-MaC heeft ons beloofd dat we direct gebruik kunnen maken van hun API. We hoeven alleen de recepten voor ons gebak vooraf aan te leveren. Zij zetten ze in het systeem en voorzien ons van de product id's. [Out of scope, je mag er dus van uitgaan dat een BMAC klant de product Id’s kent] Telkens wanneer een bestelling van onze eigen klant wordt afgerond, verzamelen we deze in ons eigen ERP systeem. Af en toe exporteert dan iemand een purchase order bestand en levert dit aan aan de BMAC API.
  + Wij halen de taarten de dag erna op en leveren ze af. We hopen snel veel bestellingen te krijgen, zodat ook wij kortingen kunnen krijgen, afhankelijk van ons loyaliteitsniveau, we starten op het bronzen niveau, wat eigenlijk nog geen korting geeft.
  + Ons unieke selling point is het feit dat je je bestelling voor een bepaald tijdstip kunt annuleren. B-MaC vertelde ons dat 22 uur het uiterste tijdstip is, dus een klant kan annuleren en opnieuw bestellen zoveel als die wilt.
  + Omdat we nieuwsgierig zijn naar hoe vaak dit gebeurt, welke taarten erg vaak worden geannuleerd, hopen we dat B-MaC ons desgevraagd de nodige informatie kan geven. We vragen ons wel af of we hun loyaliteitssysteem kunnen misleiden door een heleboel bestellingen te bestellen en te annuleren, voegde de vertegenwoordiger van Fruit Cake Inc. er gekscherend aan toe.

**Systemen**

## 

## Functionaliteit

### Clients API (API endpoints)

In onderstaande tabel zijn de interacties weergeven met de klanten API, hiervoor moeten scenarios voor geschreven worden zodat je dit kan demonstreren via http-files.

Tijdens het initiële ontwikkelproces kan de user meegegeven worden in de body, maar uiteindelijk is het wel het doel om dit via de authenticatie tokens te extraheren.

Opgelet: de ProductNumber is een identificatie die gekend moet zijn door zowel de Clients applicatie als de Bakery applicatie. De klantenbeheerder kan bepalen hoeveel een product kost.

Volgende functionaliteiten dienen voorzien te zijn.

* Als gebruiker moet ik een account kunnen aanmaken. **US-1**
* Als gebruiker moet ik een account kunnen verwijderen. **US-2**
* Als gebruiker moet ik een order kunnen aanmaken. **US-4**
  + Een order bestaat minstens uit een lijst van producten met telkens een aantal
  + Het moet mogelijk zijn om een order aan te maken als een kopie van een order uit de bestelgeschiedenis **US-5**
  + Bij het aanmaken van een order worden de details van dit order teruggegeven incl. prijsinfo. De prijsinfo bestaat minstens uit de totale kostprijs van het order met het type korting dat gegeven wordt (afhankelijk van loyaliteitslevel) en een absoluut korting in euro.
* Als gebruiker moet ik een order kunnen bevestigen. **US-6**
  + Het order wordt op dat moment, als event, doorgegeven aan de Bakery applicatie. **US-19** . Hierna kan het dus niet meer geannuleerd worden. **US-7**
* Als gebruiker moet ik (niet-bevestigde) orders kunnen annuleren. **US-7**
* Als gebruiker wil ik punten kunnen verdienen: 1 punt per 10 euro. **US-8**
  + Indien ik brons ben < 1000 punten, krijg ik geen korting
  + Indien ik zilver ben < 5000 punten, krijg ik 5% korting
  + Indien ik goud ben <= 10.000 punten, krijg ik 10% korting
  + Indien ik platina ben > 10.000 punten, krijg ik 20% korting **US-11** + **US-12**
* Als gebruiker wil ik een overzicht hebben op mijn huidige loyaliteitsniveau. **US-9**
  + Ik krijg minstens mijn huidige puntenaantal en niveau te zien.
* Als gebruiker wil ik een overzicht kunnen opvragen van mijn gemaakte bestellingen, eventueel gefilterd op "geannuleerd", op “openstaande/huidige bestellingen” en op een date range. **US-10®†**
* Als klantenbeheerder wil ik de loyaliteitsniveaus en de bijbehorende drempels en kortingen beheren. **US-13**
* Als gebruiker met grote bestellingen kan ik een bestand uploaden via een REST endpoint. Deze worden ineens als geconfirmeerd beschouwd. Deze bestelling is een zogenaamd **purchase order** en alleen mogelijk voor B2B bedrijven. **US-14**
* Als klantenbeheerder wil een nieuw product kunnen aanbieden
  + Een nieuw product komt in het systeem binnen dmv. een event dat uitgestuurd wordt door de bakkerij. Zo’n product wordt aangemaakt met een blanco prijs. **US-18**
  + Als klantenbeheerder wil ik een lijst kunnen opvragen van nieuwe (ongeprijsde) producten **US-16**
  + Als klanten beheerder wil ik een product kunnen aanbieden (= de prijs zetten van een ongeprijsd product) **US-17**

Het concept van facturatie en betaling valt buiten scope.

Een eenvoudige file voor een **purchase order** ziet er als volgt uit. Een xsd zal meegeleverd worden.

|  |
| --- |
| **<?xml version="1.0"?>**  **<PurchaseOrder ~~PurchaseOrderNumber="99503" OrderDate="2023-10-20"~~>**  **~~<Account Type="B2B">1de496e5-4486-4dec-ae2c-f62317f07c0f</Account>~~**  **<Items>**  **<Item ProductNumber="33ab0ee2-cb15-45cf-b96e-c6a08d267e8d">**  **<ProductName>Chocolade Donut</ProductName>**  **<Quantity>100</Quantity>**  **<SpecialInstruction/>**  **</Item>**  **</Items>**  **</PurchaseOrder>** |

Samenvatting van alle mogelijke interacties, deze lijst is niet exhaustief.

Hier zullen nog interacties moeten bijkomen tussen de Clients applicatie en de Bakery applicatie.

Authenticatie gebeurt via het IAM platform. Een gebruiker logt in via het IAM platform **US-3** met behulp van een email en wachtwoord en krijgt een token terug (verder aangeduid als “[User]”). Bij communicatie met de Klant API zal de gebruiker dit token moeten meegeven. **US-21** + **US-22**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event | Type | Params/Data | Who | Return |
| **create account** | Call | [User]  Type (B2B, B2C) | User | http status code  accountUUID  [User][[1]](#footnote-1) |
| **delete account** | Call | [User] | User | http status code |
| **create order** | Call | [User]  Lijst van ProductNumbers met telkens een aantal  OF  OrderNumber (UUID) van een reeds afgesloten order (=maak kopie) | User | http status code  order details incl.   * details prijs * OrderNumber (UUID) |
| **confirm order** | Call | [User]  OrderNumber (UUID) | User | http status code |
| **cancel order** | Call | [User]  OrderNumber (UUID) | User | http status code |
| **get orders** | Call | [User]  date range [optional from to]  geannulleerd JA/NEE  enkel openstaande JA/NEE | User | http status code  order detail list |
| **get loyalty info** | Call | [User] | User | http status code  loyalty level  loyalty points |
| **batch upload**  **‘purchase order’** | Call | [User]  multipart file | User | http status code  correlationUUID |
| **get batch status** | Call | [User]  correlationUUID | User | http status code  ONTVANGEN  IN VERWERKING  VERWERKT  FOUTIEF **US-15** |
| **get new products** | Call | [User] | Klanten-  beheerder | http status code  list of unpriced products |
| **patch product** | Call | [User]  ProductNumber, Price | Klanten-  beheerder | http status code |
| **order products** | Message  Destination: Bakery |  |  | **US-19** |

### Warehouse (API endpoints)

* Als magazijnbeheerder wil ik weten welke ingrediënt in het warehouse liggen. **US-33**
* Als magazijnbeheerder wil ik dat uitgaande orders automatisch ingewilligd worden en beantwoord worden, hier worden zogenaamde leveringen voor aangemaakt. **US-34**
  + Een levering bevat telkens het ingrediënt met de kwantiteit.
  + Indien aan een levering niet kan voldaan worden (onvoldoende stock), wordt dit gemeld aan de Bakery applicatie. Het plaatsen van bestellingen bij leveranciers voor het aanvullen van de stock valt buiten scope.
  + De levering wordt bevestigd voor ontvangst door de Bakery applicatie. **US-49**
* Als magazijnmanager wil ik de voorraad van ingrediënten bijwerken.**US-36**
* Als magazijnbeheerder heb ik interesse in nieuwe ingrediënten die mogelijks relevant worden voor mij.
  + Een nieuw product komt in het systeem binnen d.m.v.. een event dat uitgestuurd wordt door de bakkerij. Aan een product zijn steeds ingrediënten gekoppeld. **US-35**

Er is geen rolverdeling en authenticatie op de Warehouse API. We gaan ervan uit dat elke actie door de magazijnbeheerder wordt uitgevoerd.

|  |  |  |  |
| --- | --- | --- | --- |
| Event | Type | Params/Data | Return |
| **get-ingredients** | Call |  | http status code  list of ingredients with amount in stock |
| **patch ingredients** | Call | IngredientNumber  Stock count | http status code |
| **deliver** | Message  Destination: Bakery | OrderNumber (UUID)  Timestamp  Success / Failed |  |

### Bakery (Web Application)

Voor deze applicatie dient een UI[[2]](#footnote-2) voorzien te worden.

Een nieuw product maken is een heel proces, er zal een goede naam voor het product gekozen moeten worden en dan zal er een recept aangehangen worden. Dit recept gaat de bakker dan verfijnen totdat hij het recept finaal gemaakt heeft. Zodra dit gebeurt kan de klantenbeheerder het product opnemen en er een prijszetting voor doen, het magazijn kan de nodige ingrediënten inslaan.

* Als bakker moet ik een product kunnen aanmaken. **US-40**

Een product bestaat minstens uit een goede naam, een recept met stappenplan en een lijst met ingrediënten en hun hoeveelheden.

* Als bakker wil ik een recept kunnen finaliseren zodat het product verkocht kan worden en de ingrediënten kunnen aangeschaft worden. **US-41** [Doe dit via zogenaamde “events” op de queue]
  + Op dit moment zal er een event uitgestuurd worden (“nieuw product” incl. lijst van ingrediënten). Zowel de Clients API als de Warehouse API zijn geïnteresseerde partijen. **US-48**
* Als bakker wil ik een product kunnen wijzigen op allerlei manieren. **US-42**
  + Het recept kunnen wijzigen, meer bepaald:
    - Ingrediënten kunnen wijzigen (soort/aantal), verwijderen en toevoegen
  + Een product met een finaal recept kan niet meer gewijzigd worden.
* Als bakker wil ik op drukke dagen de mogelijkheid hebben om het bakproces voor de reeds binnengekomen orders te starten. Ik geef aan via de web applicatie wanneer ik wil starten met bakken en wanneer een product gebakken is. **US-43**
* Als bakker wil ik dat het bakproces automatisch gestart wordt op 22u. **US-44**
* Als bakker wens ik dat bij het starten van het bakproces de nodige ingrediënten met hun hoeveelheden besteld worden bij het magazijn. **US-45**
* Als bakker wens ik een overzicht van alle huidige producten. **US-46**
* Als bakker wil ik een product deactiveren, waarna het niet meer gebakken of besteld kan worden. **US-47**

Bovenstaande feature-lijst dient als web-applicatie (MVC/Thymeleaf) geimplementeerd te worden.

Hieronder staan de events (enkel messages) die het systeem uitstuurt.

|  |  |  |
| --- | --- | --- |
| Event | Type | Params/Data |
| **new product** | Message  Destination: Warehouse, Klant | ProductNumber + name + list<Ingredients> (name + amount) |
| **deactivate product** | Message  Destination: ~~Warehouse~~, Klant | ProductNumber **US-20** + **US-47** |
| **order ingredients** | Message  Destination: Warehouse | OrderNumber (UUID)  timestamp  IngredientNumber (UUID),  amount |

## User Stories

De totale lijst van user stories is meegeleverd in een csv bestand, dit kan je importeren in GitLab en hiervoor worden dan issues aangemaakt. Op basis van deze issues tracken we je vooruitgang.

## Enkele concepten in detail:

* **Klant**: is altijd een persoon die een bedrijf vertegenwoordigt of zichzelf. Het bedrijf heeft een gebruiker met als specifieke functie om dit te doen. Bij het aanmaken van een account wordt aangegeven of het om een B2B, of B2C klant gaat.
* **Klantenkaart:** we hebben klantenkaarten met 4 niveaus, brons, zilver, goud en platina.   
  Brons is het instapniveau en geeft geen korting, zilver geeft 5%, goud 10% en platina geeft maximaal 20% volumekorting.   
  Je kunt loyaliteitsbeloningen krijgen per bestelling. 10 euro geeft je 1 punt. Zodra je de grens van het niveau van je klantenkaart bereikt, ga je een niveau omhoog.  
   Een bestelling kan uit 1 of meer producten bestaan.
* **Product**: is een eindproduct dat in de bakkerij kan worden gebakken en direct aan klanten kan worden verkocht. Elk product heeft een recept met een lijst van ingrediënten en de benodigde hoeveelheid. Er worden voorlopig geen ingrediënten in de bakkerij bewaard, elk ingrediënt wordt direct uit het magazijn besteld. Zodra de benodigde ingrediënten voor een product in de bakkerij aanwezig zijn, wordt het product direct gebakken.
* **Een inkomende order:** is een order naar een externe leverancier voor ingrediënten... als de leverancier de order kan uitvoeren, ontvangt het magazijn 1 of meer zendingen tbv de order. [Inkomende orders alleen in scope te nemen indien de andere scope klaar is]
* **Een uitgaande order:** is een order om ingrediënten van het magazijn naar de bakkerij te krijgen. Als het magazijn de order kan uitvoeren, zal het magazijn een levering naar de bakkerij maken. Als het magazijn niet genoeg voorraad heeft, wordt er een inkomend order aangemaakt om de voorraad aan te vullen. Uitgaande orders worden eerst uitgevoerd voordat ingrediënten op voorraad worden gelegd. [Inkomende orders alleen in scope te nemen indien de andere scope klaar is, je mag er ook van uitgaan dat er oneindige voorraad is, om de levering aan te maken bouw je desnoods een delay in]
* **Een orderbatch:** bestellingen van klanten worden opgeslagen in wat in de bakkerij een batch wordt genoemd. Dat is de batch waar de bakkerij zich aan kan verbinden voor de volgende dag. Er is echter een grens, omdat de bakkerij niet meer kan garanderen dat alle producten de volgende dag gebakken zullen zijn, deze is momenteel gezet op 22u. Alle ingrediënten voor die specifieke batch worden in één keer besteld en resulteren dus in één uitgaande order.
* **Een levering:** elke uitgaand order zal een corresponderende levering hebben uit het magazijn. Zodra alles geleverd is aan de bakkerij kan het product gebakken worden. [Zodra de levering binnen is, mag je de nodige producten als gebakken beschouwen]

## Voorzien zijn op mogelijke wijziging en uitbreiding

De code moet (met behulp van interfaces, patterns,...) open zijn voor toekomstige uitbreidingen **US-27** + **US-37** + **US-50**

* Andere loyalty levels en eventueel een extern COTS-systeem[[3]](#footnote-3) om dit te beheren.
* Er is reeds gepeild door het management of we ons magazijn zouden willen openstellen voor ingrediënten van andere bakkerijen. We denken aan uitbreiding over heel het land.
* Grote afnemers van onze diensten zouden graag via asynchrone messaging willen werken zodat ze hun bestellingsysteem aan dat van ons kunnen koppelen ipv telkens de API moeten oproepen met hun aankooporders.
* Warehouse API: Als magazijnbeheerder wil ik mijn magazijn kunnen indelen in verschillende secties die verschillende kenmerken aankunnen van ingrediënten. **US-30**
  + Je zou kunnen starten door volgende kenmerken te voorzien:
    - 3 verschillende temperatuurzones: BEVROREN, GEKOELD, GEMATIGD
    - 3 verschillende houdbaarheidszones: TGT (te gebruiken tot), THT (tenminste houdbaar tot), ZONDER
  + *Opgelet: een ingrediënt kan eender welke combinatie van deze kenmerken hebben. Een ingrediënt zonder houdbaarheidsdatum (denk aan een pompoen), kan zowel bevroren, gekoeld of gematigd bewaard worden. Met de huidige kenmerken zullen we dus al 9 secties moeten aanmaken.*
  + Unit testen ivm het toewijzen van ingrediënten aan hun respectievelijke secties
* Bij inspectie van de brandveiligheid is ontdekt dat we verschillende bluszones moeten voorzien, oliën en vetten moeten in een schuimzone bewaard worden. Elk ingrediënt zou dan een gevaarklasse als extra kenmerk krijgen.
* Andere data formaten en protocollen (JSON, XML, verschillende soorten queues, …)
* Als gebruiker wil ik meldingen ontvangen wanneer mijn bestelling klaar is. **US-31**
* Als beheerder wil ik verkooprapporten genereren op datum, product of gebruiker. **US-32**

## Testen

Volgende geautomatiseerde testen worden minstens voorzien op de verschillende applicaties:

* Unit testen voor de prijsberekening van het order **US-23**
* Unit testen i.v.m. loyaliteitsniveaus **US-24**
* Integratie testen vanuit de controllers voor de Clients API **US-25**

We verwachten een code **line coverage** van minstens 60% op de Clients applicatie. **US-26**

## Configuratie en Logging

Wat zeker geconfigureerd[[4]](#footnote-4) moet kunnen worden: **US-28** + **US-38** + **US-51**

* Infrastructuur connecties
  + Databases
  + Queues
  + IAM, IDP configuraties
  + Andere…
* Business specifieke configuratie:
  + Cut off moment (22u momenteel)
  + Loyaliteit niveaus:
    - Punten plafonds
    - Korting percentages
  + Andere…

De code wordt voorzien van zinvolle logging statements zowel op vlak van diagnose (debug, info) als foutafhandeling (warn, error...). Zaken die kunnen mislopen en dus gelogd moeten worden zijn onder meer: **US-29** + **US-39** + **US-52**

* De API wordt aangeroepen met foutieve input data (bv. niet gekend order of bij het valideren van een business rule, denk aan het annuleren van een reeds bevestigde order).   
  Een error code (volgens REST conventies) wordt teruggestuurd en een warning gelogd.
* Databases kunnen niet bereikt worden of geven onverwachte fouten terug. Een error wordt gelogd en een error code teruggestuurd (indien de database toegang het gevolg was van een API call).
* Queues kunnen niet bereikt worden of geven fouten terug*.* Een error wordt gelogd. Indien het de uitgaande order topic betreft wordt deze later automatisch opnieuw verwerkt.
* Xml file kan niet correct ingelezen of verwerkt worden. De file wordt niet verwerkt. Een error wordt gelogd

## Security

De REST endpoints van de Klanten applicatie dienen minimaal beveiligd te worden volgens de regels van de kunst. Kies hiervoor een authentication grant type waarvan je de voordelen en nadelen begrijpt. Er zal een introductiesessie omtrent deze materie gegeven worden.

<https://datatracker.ietf.org/doc/html/rfc6749>

## Technische specificaties

Je gebruikt **Spring Boot** (Java of Kotlin) en Gradle of Maven voor de ontwikkeling.

Je gebruikt een Postgres database, je mag voor elke applicatie dezelfde database gebruiken. Echter gaan we werken volgens de principes van een microservice architectuur. Elke applicatie (of [Software Architecture 23-24 The Bakery - Google Docs](https://docs.google.com/document/d/14hGHJxErAcPLOx3-JgEPKWiySBL4yldmqjSrkNcM5mI/edit)microservice) beschikt over een eigen datastore. Met Postgres kan je dit opzetten door gebruik te maken van schema’s.

Benoem deze schema’s als volgt: Clients, Bakery, Warehouse  
<https://www.postgresql.org/docs/current/ddl-schemas.html#DDL-SCHEMAS-CREATE>

Je kan zelf een postgres database opzetten in een docker-compose. Maak een aparte infrastructuur folder in je project met deze informatie. Let op dat je je datafiles niet mee in Git stopt, dit kan je doen door een .gitignore file aan te maken.

Voor de queue gebruik je CloudAMQP messaging. (<https://www.cloudamqp.com/>)

Je mag echter ook deze in een docker plaatsen. Als je gebruik maakt van de docker-compose file en je wenst de cloud oplossing te gebruiken, kan je de service best verwijderen uit de docker-compose file.

Als IAM en IDP gaan we Keycloak gebruiken. Je mag net zoals de RabbitMQ een cloud oplossing gebruiken (<https://www.cloud-iam.com/>) of deze in een docker draaien.

Als je gebruik maakt van de docker-compose file en je wenst de cloud oplossing te gebruiken, kan je de idp service met de idp databank best verwijderen uit de docker-compose file.

In bijlage is een mogelijke docker-compose.yml meegeleverd, deze spint een volledige omgeving op.

Het versiebeheer gebeurt in **GitLab.** Ookde project planning en opvolging gebeurt aan de hand van **GitLab**.

**Zie de slides *Inleiding* en de infosessies voor verdere details over de opvolging, technologieën, architectuur etc..**

## Bijlage infrastructuur bestand docker-compose

Er wordt ook een docker-compose bestand meegeleverd dat je naar believen mag gebruiken en aanpassen.

Het docker-bestand bestaat uit de volgende services:

* **bmac\_postgres:** Een database voor je applicatie:

Dit is een postgres database. Gebruik deze database voor je applicatie. De postgres databasepoort wordt toegewezen aan **5432**, controleer de service postgres\_bmac\_db voor meer informatie over hoe je verbinding maakt met welke referenties.   
Volumes worden gemapt naar de hostmachine onder de ./bmac/postgres folder, zorg ervoor dat je ze niet mee commit naar git door een .gitignore bestand in te stellen, zodat hij deze folder negeert.

Deze service maakt verbinding via een aangepast docker-netwerk genaamd backend.

* **bmac\_rabbitmq:** Een berichtgeoriënteerde middleware:

Dit is rabbitmq, de beheerpoort is toegewezen aan poort **15672**, controleer de service bmac\_rabbitmq voor meer info ivm username en paswoord.

De AMQP poort zelf is **5672**

Volumes worden gemapt naar de hostmachine onder de ./bmac/rabbitmq, zorg ervoor dat je ze negeert in je .gitignore bestand. Deze service maakt verbinding via een aangepast docker netwerk genaamd backend.

Opgelet: linux systemen doen moeilijk als je je rechten niet goed zet. Kijk naar de logging indien er iets niet goed opstart en pas de rechten aan waar nodig.

* **idp\_keycloak:** Een identiteitsprovider:

Dit is een keycloak identity provider. Gedetailleerde documentatie is hier te vinden. <https://www.keycloak.org>

Deze service heet idp\_keycloak en tijdens de colleges zal duidelijk worden hoe we met deze service gaan integreren. Er wordt een speciaal docker-netwerk opgezet met de naam kc.

De beheerconsole staat op [http://localhost:8180/](http://localhost:8180/auth/)

Ook een mooi uitgangspunt voor documentatie.

* **idp\_postgres**: Een postgres database voor alles wat met keycloak te maken heeft.

Dit is ook een postgres database, maar is niet geschikt voor directe interacties. Het linkt naar een ander docker netwerk speciaal voor keycloak genaamd kc network.

Net zoals de andere postgres, is er een volume gemapt op de hostmachine, dit onder ./bmac/idp, voeg deze ook zeker toe aan de .gitignore file.

1. [User] staat voor een security token waarmee gebruikers zich kunnen authenticeren [↑](#footnote-ref-1)
2. Via Spring MVC met Thymeleaf templates [↑](#footnote-ref-2)
3. Commercial off-the-shelf: pakket-software [↑](#footnote-ref-3)
4. application.properties in Spring terminologie [↑](#footnote-ref-4)