Een flexibele architectuur voor zaaksystemen

Versie	Wijzigingen
1.0	Eerste versie
1.1	Opmerkingen verwerkt van Gevik Babakhani, Chris Widdows en Corné Smiesing

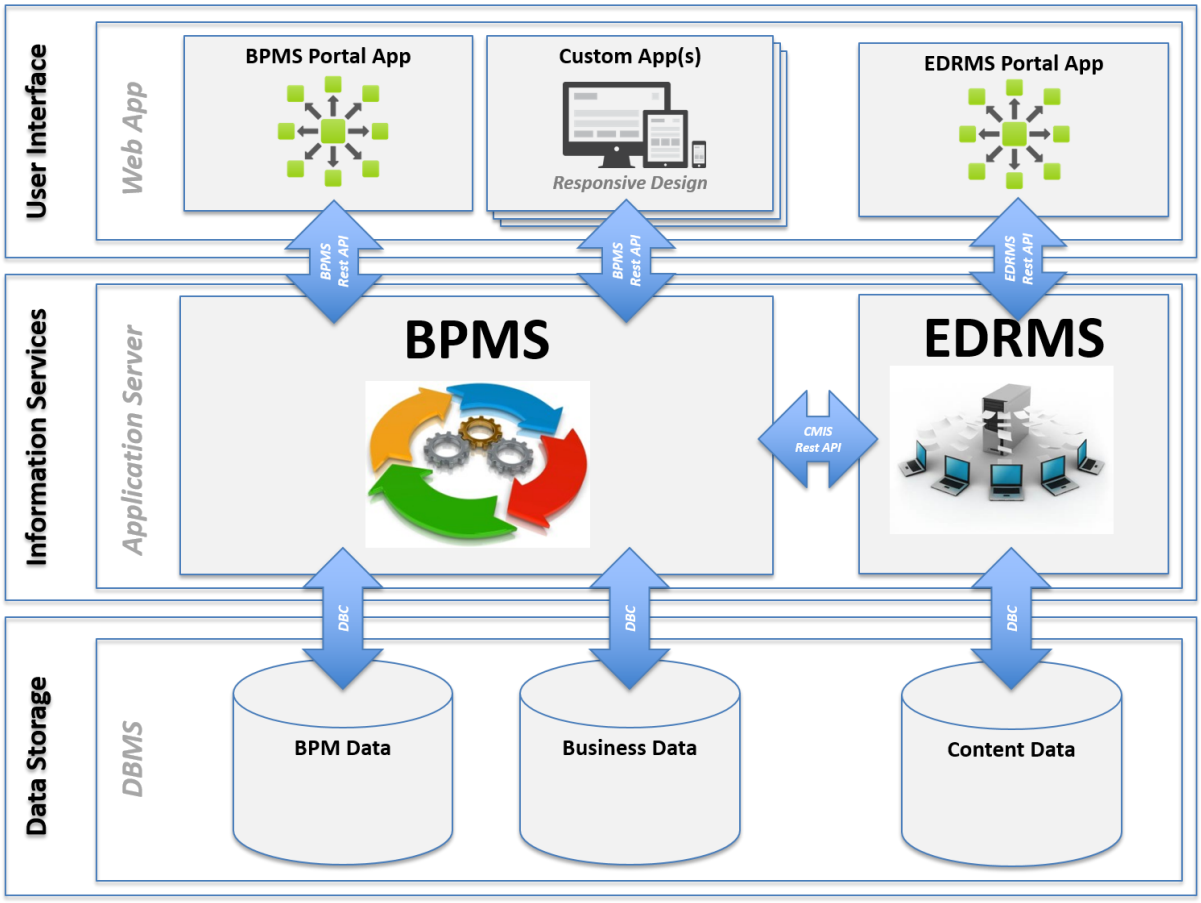
Een flexibele architectuur

Deze whitepaper beschrijft een flexibele architectuur voor zaaksystemen die zowel gebruikt kan worden voor een closed source als een open source architectuur.

Om te komen tot een optimale architectuur is gekeken naar de verschillende functionele blokken die onderscheiden kunnen worden in een zaaksysteemomgeving. Een zaaksysteem heeft een aantal grote functionele componenten, zoals een gebruikersinterface, BPM-functionaliteit, document-management-functionaliteit en een database management systeem.

Een basisarchitectuur voor een zaaksysteemomgeving

Onderstaande architectuur geeft op een flexibele manier invulling aan een zaaksysteem-omgeving.



De eerste indeling die gemaakt is in de structuur is een tamelijk traditionele lagenstructuur:

1. **User interface**. Binnen deze lag valt alle functionaliteit die beschikbaar is voor gebruikers, ook die voor superusers die (functioneel) beheer doen op de het Business Process Management Systeem (BPMS) of het Electronic Document and Records Management Systeem (EDRMS).
2. **Information services**. De laag Information services draagt zorg voor enerzijds de verwerking van alle gebruikersacties naar de gegevensopslag. Bovendien zorgt deze voor de afhandeling van signaleringen op basis van configuratie en gegevens.
3. **Data storage**. In de data storage vindt de opslag van alle gegevens plaats. Dit betreft niet alleen gebruikersgegevens en documenten, maar ook metagegevens die nodig zijn voor indexering van document en metagegevens van het proces.

Ieder van die componenten kan verder opgedeeld worden.

Uitdieping basisarchitectuur

Ieder component in deze architectuur kan los ingevuld worden. Hierbij is wel belangrijk dat gekozen wordt voor software die een goede ondersteuning biedt van open standaarden. Open standaarden zijn de basis voor de flexibiliteit van deze architectuur. Indien software die ondersteuning niet biedt, kan die alleen tegen grote inspanning ingepast worden in de architectuur. Het regelmatig aanpassen van functionaliteit zal dan snel tot hoge exploitatiekosten leiden.

User interface

De laag voor user interface bestaat uit drie componenten:

1. **Custom app(s)**. De meeste gebruikers krijgen toegang via een gespecialiseerde webapp. Dit is de feitelijk interface van het zaaksysteem naar de gebruiker toe. Vanwege ontwikkelen als Bring Your Own Device (BYOD) en Het Nieuwe Werken (HNW) is het van groot belang dat deze functioneert op veel verschillende apparaten en systemen (telefoon, tablet, laptop, pc). Wij maken daarom gebruik van Responsive Design, dat bedoeld is om zich aan te passen aan het device waarop het gebruikt wordt.
   Het is mogelijk een monolithisch zaaksysteem aan te leggen, maar de flexibiliteit van de architectuur kan ook gebruikt worden om voor verschillende gebruikers-perspectieven (rollen of autorisatieniveaus) verschillende apps te maken. Ook onderscheid naar devices ligt voor de hand. Hoewel responsive design zich naadloos aanpast, ligt grootschalige gegevensinvoer niet voor de hand voor mobiel. Het is de kunst van het ontwerp om ervoor te zorgen dat enerzijds ieder van de devices optimaal gebruikt wordt, zonder dat dat anderzijds leidt tot het opzetten van specifieke apps voor devices.
2. **BPMS-portal app**. Het BPMS-portal geeft rechtstreeks toegang tot het BPM-systeem en zal primair gebruikt worden door supergebruikers of functioneel beheerders. Hier bevindt zich geen functionaliteit voor het behandelen van zaken, maar alleen functionaliteit voor ondersteuning van het zaaksysteem zelf.
3. **EDRMS-portal app**. Ook het EDRMS-portal is primair bedoeld voor functioneel beheerders en geeft rechtstreeks toegang tot het EDRMS.
   In sommige gevallen kan ervoor gekozen te worden de EDRMS-portal ook open te stellen voor reguliere gebruikers, met als doel documenten rechtstreeks te ontsluiten.

Het geniet echter de voorkeur dat dit soort functionaliteit vanuit het zaaksysteem wordt geboden (via de custom webapp).

Information services

De laag voor Information services heeft twee componenten:

1. **BPM-systeem**. Het BPM-systeem is verantwoordelijk voor het tot een keten rijgen van activiteiten (taken) van medewerkers. Zaken als routering van taken, status- en voortgangsbewaking en signalering liggen bij het BPMS.
   Zaaktypen en de daarbij horende workflows worden in het BPMS ingericht.
   Bovendien worden relevante documenttypen gekoppeld, op basis van de documenttypen die gedefinieerd worden in het DMS.
2. **EDRMS**. Het versie-management, de archivering en het beschikbaar stellen van alle documenten en records wordt door het EDRMS verzorgd. Ook het bewaken van bewaartermijnen kan neergelegd worden bij het EDRMS.
   Documenttypen en records en de daarbij horende tags en bewaartermijnen worden in het EDRMS ingericht.
   NB Vaak bieden EDRMS'en een vorm van workflow aan. Deze functionaliteit is over het algemeen beperkt en biedt niet de mogelijkheden van een BPMS.

De Information Services worden doorgaans centraal in de (lokale) Cloudomgeving beschikbaar gesteld. Technisch gezien moet het mogelijk zijn deze lokaal te installeren voor bijvoorbeeld ontwikkel- en testdoeleinden, maar het geniet de voorkeur hiervoor een standalone-server te gebruiken.

Data storage

De laag voor data storage heeft drie databases:

1. **BPM data**. Deze database bevat alle gegevens die door het BPMS gebruikt worden voor de administratie van werkzaamheden en taken. Hierbij kan gedacht worden aan status- en voortgangsinformatie, workflows en andere processysteemgegevens.
   Deze gegevens worden door het BPMS gewijzigd en normaal gesproken niet door gebruikers.
   Deze gegevens hebben primair een metafunctie; ze zijn bedoeld voor de besturing van het proces. Een deel van deze informatie zal echter wel gebruikt worden voor statusinformatie en managementrapportages; veel informatie over tijdigheid en kwaliteit ligt opgesloten in deze metagegevens.
2. **Business data**. Business data zijn gegevens die direct door gebruikers zelf worden gemuteerd. Deze gegevens beschrijven de toestand van objecten die vanuit bedrijfsperspectief zijn gemodelleerd (denk aan locaties, producten, klanten, leveranciers en financiële transacties).
3. **Content data**. Content-data zijn alle gegevens die door het EDRMS gebruikt worden voor de administratie van documenten en records, plus de documenten en records zelf. Tags die bij een document opgeslagen worden zijn van groot belang, die vormen de basis voor het categoriseren en terugvinden van documenten.
   NB Als gebruik gemaakt wordt van workflow-functionaliteit in het EDRMS, worden ook die gegevens hier opgeslagen. Wij gaan er echter in de architectuur vanuit dat een op zich staand BPMS wordt gebruikt.

Interfaces

De communicatie tussen de lagen in de architectuur moet plaatsvinden op basis van open standaarden, alleen op die manier kan de flexibiliteit van die architectuur bestaan:

1. **Rest API's**. Communicatie vanuit de user interface naar de Information services laag wordt via rest API's afgehandeld, een algemeen geaccepteerde standaard voor uitwisseling tussen verschillende computersystemen.
2. **DBC**. Communicatie tussen de Information services en data storage lagen vindt plaats door middel van een DBC-variant die aansluit bij de omgeving, zoals ODBC of JDBC. Hierbij moet een balans gezocht worden tussen de mogelijkheden van het DBMS (native SQL kan een struikelblok zijn), de gebruikte tooling en open standaarden.
3. **CMIS**. CMIS is een open standaard bedoeld voor het uitwisselen van gegevens met content management systemen, waaronder ook EDRMS'en vallen. Wij gebruiken hiervan een Rest API-implementatie.

NB Er vindt geen directe communicatie plaats tussen de user interface en de data-storage-laag. Communicatie tussen deze twee lagen vindt altijd plaats via de information-management-laag.

Verdere ontwikkelingen

Hoewel de hier beschreven architectuur veel flexibeler en compacter is dan traditionele architecturen, is het in basis toch een monolithische architectuur, aangezien die nog steeds grote componenten bevat die verschillende functionaliteiten in zich hebben.
In een toekomstige paper zullen we kijken naar een architectuur die meer gebaseerd is op microservices, veel kleinere functionele eenheden, die met elkaar verbonden zijn en zo systemen vormen. Dit leidt tot een verdere flexibilisering van functionaliteit en herbruikbaarheid van componenten.

Roberto Lambooy/Onno Haldar, ZaakgerichtWerken.nu