Komponent oversigt

Jesper Toplund

Contents

1 Formål

Det primære formål med rapporten er at opridse et roadmap for hvad vi bør fokusere på, hvis vi gerne vil opnå en sammenhængende, tidssvarende og vedligeholdbar arkitektur.

2 Omfang

Denne rapport er skrevet ud fra information i DR-systemkatalog og de betragtninger der er blevet gjort i forbindelse med arkitekturrapporten udarbejdet i samarbejde med Eksponent. Estimater og overslag, der er angivet i rapporten, er foretaget ud fra et kvalificeret bud på besvær og omkostninger og ikke ud fra en detaljeret analyse af hvert enkelt projekt. Mere præcise estimater vil være mulige for hvert delopgave, men er uden for scope af denne rapport.

Mere præcise estimater vil være mulige for hvert delopgave, men er uden for scope af denne rapport.

3 Executive summary

Arkitekturen i Web & Apps bærer præg af at have vokset organisk frem. Når et forretningsproblem har vist sig, er en løsning på problemet lavet ud fra de for hånden værende ressourcer og kompetencer, uden hensyntagen til hvordan løsningen passer ind i resten af systemlandskabet. Det har resulteret i at vi i dag har mange eksempler på parallelle implementeringer, på systemer der ikke kan vedligeholdes, på ukendt aftager landskab og på fragmenteret infrastruktur.

Hvis vi er interesseret i at løfte infrastrukturen fra hvad vi har i dag til et niveau som er lettere at vedligeholde, hurtigere at udvikle til og sikrere at teste, så er der en række tiltag vi bør gennemføre.

De enkelte skridt ridses op i de følgende afsnit men kan kort beskrives som: De vigtigste skridt vi skal igennem er følgende:

- Design af gennemgående datamodel
- Konsolidering af infrastruktur
- Omskrivning af forældede produkter
- Sikre ensartet DevOps fundament i afdelingen

Vi skulle gerne ende op med en arkitektur, hvor det er muligt at have styr på vores aftager-landskab. Vi skal også have mulighed for hurtig kommunikation imellem interne komponenter via sikret og afgrænset netværk. Det skal være muligt at teste komponenter individuelt og i et sammenhængende produktionslignende test setup.

Vi kan passende gå i gang med de lavt hængende frugter i form af at flytte de letteste kandidater over på pipeline, imens vi går i gang med de designtunge dele af opgaven (f.eks. design af fælles datamodel og redesign af relationssystem).

Det er svært at tids estimere præcist uden at grave dybere ned i detaljerne for de enkelte opgaver. Selve transformationen vil formentligt tage 2 år eller længere. Hvis vi i den tidsperiode stadig vil kunne varetage normale udviklingsopgaver, så bør vil allerede nu overveje at mande op i de forskellige teams.

4 Forretningsbehov

Vi har en forpligtigelse i at tænke forretningsbehovet for DR som helhed ind i de beslutninger vi tager over for Roadmap.

Alle ændringer og beslutninger vi tager burde gerne føre os i retning af en bedre, hurtigere og mere stabil platform. Dermed bør vi i alle beslutninger kunne beskrive hvordan beslutningen taler ind i bringe os i retning af vores referencearkitektur.

4.1 Nyhedsapp

Nyhedsappen har fungeret fint i 4 år og er i efteråret 2019 blevet vurderet af app bureauet Nodes som værende så godt optimeret som det er muligt baseret på den teknologi den er bygget på. Hvis nyhedsappen skal fortsætte med at være tidssvarende og blive ved med at give en god brugeroplevelse, så var det Nodes anbefaling at skifte over til en native applikation. Det vil sige at der skal udvikles en native IOS og en native Android version.

Hvis vi skal have en bedre brugeroplevelse eller markante udvidelser af nyhedsappen, så skal vi efter alt at dømme bygge appen helt op igen fra bunden af i en Native udgave. I den forbindelse vil det være meget relevant at vi også sikre datagrundlaget og platformen under appen, så vi kan få den bedst mulige brugeroplevelse. Hermed vil det være relevant at sikre datamodel, event system og relations system er forberedt på denne udvikling.

4.2 Drupal 7 pensioneres

Drupal 7 (og Drupal 8) står over for End Of Life i november 2021. Inden da skal vi have taget stilling til hvordan vi skal fortsætte derfra. Oprindeligt var Drupal hele vores udgivelses platform og det styrende system for alt indhold på DR.dk. I den rejse som DR.dk har været på de sidste par år, er flere og flere af Drupals oprindelige ansvarsområder blevet flyttet ud af Drupal og over i andre systemer. Dermed står vi med markant andre behov i forhold til CMS system end vi gjorde da vi oprindeligt valgte Drupal 7. Det er ikke nødvendigvis en direkte udskiftning af Drupal 7 med Drupal 9 der giver mest mening.

For at sikre at vi kan tage det bedst mulige valg over for denne udskiftning skal vi sikre at vores Datamodel er på plads og vi har afdækket API og event behov i forhold til redaktør grænsefladen og artikel database.

4.3 Relations system

Garnnøgle og TAG system har af flere omgange været oppe og vende som problematiske systemer. Garnnøgle er i en kritisk tilstand hvor det er stort set umuligt at vedligeholde eller videreudvikle på. Ud over Garnnøgle og Tag, så er der også en række andre systemer der varetager relationshåndtering i større eller mindre omfang.

Da relationssystemer både har stor forretningsmæssig potentiale og er i kritisk vedligeholdelsestilstand, så er det et oplagt område at prioritere i vores systemgennemgang.

4.4 Udgivelseshastighed

En stor del af den kompleksitet der er i vores platform i dag er på grund af afhængighederne imellem systemerne. Hver gang der er forbindelse mellem to systemer hvor det afhænger af en brugerhandling eller et push af informationer fra et system til et andet system, så er der mulighed for forbedringer både i form af reaktionstid men også i form af systemkompleksitet ved at vende afhængighederne om og lade alle systemer udstille generaliserede APIer og lade systemer udgive og abonnere på events i forbindelse med opdateringer.

I dag, når en artikel udgives i Drupal, så skabes der et event, som opfanges af mimer og bruges til at igangsætte indekseringen. Hvis vi raffinerer den løsning og benytter et fornuftigt besked framework, så kan den samme besked benyttes til at igangsætte alle de andre tjenester rundt omkring, som også skal være på plads for at en artikel er søgbar på dr.dk.

Disse events og lignende events fra andre dele af systemet kan benyttes til at lade opdateringerne propagere ud til alle dele af systemet, inklusiv at opdatere cache lag, forside lister, personaliserings søgninger, markering af Breaking artikler, push af notifikationer til Nyhedsapp osv.

Et publiceringsevent vil dermed selv kunne triggere indeksering fra Cxense, opdatering af cache, opdatering af Garnnøgle og lignende. Det vil effektivt få ventetiden på opdatering ned i nærheden af hvad de enkelte systemer kan præstere. Ligeledes vil en eventbaseret arkitektur sikre at vi ikke behøver at pushe fra et system til et andet, men kan nøjes med at udstille APIer så et system ikke behøver at kende sine aftagere, blot at overholde den aftalte datakontrakt.

5 Vision

Web og Apps har i de sidste år gennemgået en betydelig omstrukturering af deres miljøer og services. Det landskab vi står med i dag er i langt højere grad forberedt på at håndtere de skiftende krav til danskernes medieforbrug. Der er blevet foretaget en række gode valg i opdeling af ansvar og løsere kobling imellem systemer. Denne transformation er dog foregået med begrænset koordination ud fra de behov der opstod undervejs.

Det efterlader os i dag med en arkitektur af en noget organisk karakter, hvor mange design valg er taget med et specifikt produkt for øje i stedet for at gennemtænke hvordan platformen i sin helhed kan få glæde af og inkorporere en given funktionalitet.

Det er svært at spå om præcist hvad fremtiden vil bringe af udfordringer og muligheder, det er dog muligt at spå om hvilken funktionaliteter der vil gøre det muligt for os at møde den.

5.1 Cloud strategi

DR har i årevis gået efter en Cloud strategi. Den har vi i DR.dk efterlevet efter bedste evne i den enkelte teams. Resultatet heraf er at der i hvert team er blevet taget en række valg om pipeline, hosting og platform. De enkelte teams løsning af problemet har dermed resulteret i 4 forskellige skyer og release processer. Den manglende ensretning betyder at der er noget spild arbejde i form af at vedligeholde de 4 forskellige byg og deploy pipelines men især også at der er nogle uhensigtsmæssige afhængigheder på tværs af disse skyer. Alle 4 skyer skal i spil når en ny artikel skal udgives og et udfald hos hvilken som helst af de 4 udbydere betyder at et led i kæden knækker og dermed at artikler ikke kan udkomme. Vi udnytter dermed ikke de fordele der kan være ved at have vores hosting spredt ud, men vi får ulemperne ved det.

- Offentligt tilgængelige endpoints der kun burde benyttes til intern kommunikation
- Betaling for trafik mellem sky udbydere
- Manglende ensretning af den underliggende platform
- Ekstra latenstid imellem komponenter

Vi bør konsolidere vores cloud strategi til at benytte den samme udbyder på tværs af teams og gerne også en fælles pipeline. Dermed vil vi opnå både en ensartet release og deploy process og vi vil samtidigt fælles kunne høste de forbedringer som et fælles miljø vil kunne tilbyde.

5.2 Antallet af systemer der aftager artikler vil efter alt at dømme stige

Vi bør derfor sørge for at adskille visningslag fra artikel lag i videst muligt omfang og sikre en konsistent datastruktur for artikeldata. Hvis vi får etableret en gennemgribende datamodel, så vil den samme platform kunne levere artikel data til hvilket som helst medie (web, IOS / Android, nyhedsbreve, RSS feeds, TTV osv.) der måtte forbruge det og vi vil stadig være i stand til at give en rimeligt konsistent oplevelse på tværs af medier.

Vores reference arkitektur bør i videst muligt omfang være forberedt på at vi ikke nødvendigvis kender aftager landskabet i morgen, så jo bedre vi er i stand til at udstille vores indholdsdata i en veldefineret format på så mange niveauer som muligt, jo bedre vil vi være i stand til at imødekomme ønskerne fra morgendagens aftagere, om det så er interne eller eksterne aftagere.

5.3 Event system

Vi bør anvende et Eventsystem med et fornuftig interface, respektable service kontrakter og som kan sikre hurtig besked håndtering i mellem systemerne.

For at holde os til en enkelt cloud udbyder og ikke eksponere os for øget responstider og risiko, så bør vi kikke på om vi kan benytte googles "Cloud pub/sub" eller om der er et andet system der vil passe bedre.

5.4 Behovet for personalisering vil blive endnu højere

Både forbrugere og redaktører stiller større og større krav til at vi kan vise de mest relevante artikler for en given bruger.

Et større niveau af personalisering end hvad vi har i dag vil betyde at et større antal sider skal være dynamisk opbygget baseret på personaliserede lister af hvad der skal vises hvor. Det vil ændre på hvad vi kan cache hvor og i hvilket omfang, og dermed også på belastningsniveauet på de påvirkede systemer. Content aggregation og den udstillede dataformat vil der skulle lægges en del arbejde i, lige som der også skal kikkes på både relations system og på de både interne og eksterne personaliserings systemer vi har.

Cxense er et godt eksempel på et kraftfuldt eksternt system, som vi ikke benytter til fulde endnu. Vi benytter f.eks. ikke de APIer Cxense udstiller til at sikre rettidig indeksering af publicerede sider, lige som vi ikke relaterer brugerkonti med Cxense brugerID i forbindelse med udsendelse af nyhedsbreve.

Mulighederne for at lave et moderne, personaliseret og smidigt design ligger lige for, især hvis vi sørger for at byggeklodserne der er påkrævet også er til stede.

5.5 Afdækning af trends i forbrugsmønstre vil blive mere og mere relevant

Det er vigtigt at vi holder for øje at det bør være muligt at holde styr hos forbrugsmønsteret hos vores brugere. Jo bedre vi kan spore forbrugsmønsteret på tværs af platformen, jo bedre vil vi være i stand til at imødekomme fremtidens behov og dokumentere i hvilket omfang vi opfylder de forventninger vi har til platformen. Vi skal med GDPR reglementet for øje have sikret at vi opfanger det relevante data der skal til for at kunne gøre vores platform så brugervenlig som mulig.

5.6 Større behov for at kunne linke på tværs af platforme

Det har længe været et ønske at kunne binde Radio, TV og Nyhedsartikler tættere sammen end nu. Især set ud fra personaliserings synspunkt kan det være meget relevant at kunne relatere nyhedsartikler med enten on demand radio eller video segmenter. Hvad end behovet vil være, så bør vi være sikre på at vi er forberedt på at understøtte disse relationer, både i form af adfærdsanalyser på tværs af platforme og også i form af relationssystemer der kan linke radio, video og artikler.

6 Generelt forarbejde

6.1 Indholdsdatamodel

Noget af det vigtigste for at få arkitekturen til at hænge sammen, er at vi etablerer en generel og gennemgående datamodel, som kan være med til at sikre at alle vores komponenter kan kommunikere på tværs. Det er et vigtigt stykke forarbejde, som både bidrager til muligheden for at rydde op i den nuværende hårknude-arkitektur, og gør det muligt at sikre at nyudviklede komponenter kan benyttes på tværs. Det kommer til at kræve en del arbejde at få etableret denne datamodel og derefter at få de forskellige systemer tilpasset til denne datamodel.

Vi kan enten starte helt fra bunden af og skabe en datamodel som vi forventer kan løse alle de problemer vi kan forudse eller vi kan læne os op ad en af de datamodeller der allerede er i brug i dag et sted i arkitekturen. Starter vi modelleringen ud fra en eksisterende datamodel, vil vi være i stand fra at starte fra et punkt hvor mindst 2 dele af arkitekturen allerede er alignet med datamodellen og vil være i stand til at foretage en inkrementel udvidelse / omskrivning, så den bliver mere dækkende og kan spredes ud til resten af arkitekturen. Derimod, hvis vi starter fra bunden af, så kan vi formentligt designe en renere model der lever op til vores ønsker uden at være besværet af unødvendig kompleksitet, men design heraf vil tage mærkbart længere tid og vil ikke være forankret i noget system før vi også redesigner dem.

Det anbefales at vi starter med at udarbejde en så dækkende model som muligt. Her kan vi passende tage udgangspunkt i den datamodel der allerede benyttes af Steffi og afdække hvorvidt den dækker de behov vi kan forudse. Dermed får vi et centralt udgangspunkt som, når det bredes ud, kan bruges som samlingspunkt om vores content aggregation service.

Datamodellen skal ikke nødvendigvis implementeres på tværs af hele dr.dk fra starten af, men det vil være til stor gavn hvis vi kan arbejde os i retning af en fælles datamodel i alle fremtidige større ændringer af systemerne, samt i al større nyudvikling. Datamodellen skal dermed designes ud fra et synspunkt om at den skal være fælles for så store dele af arkitekturen som muligt.

Udarbejdelse af datamodellen bør ske inden eller i forbindelse med næste store ændring i et af de centrale indholdsbærende datalag (Drupal, Mimer, Steffi, Hydra, Nyhedsapp eller lignende).

6.2 URN / URL

Vi bør i forbindelse med arbejdet på datamodellen også kikke på at få etableret en URN / URL der unikt kan identificere et hvilket som helst stykke indhold på tværs af systemlandskabet. Denne URN skal være i stand til unikt og konsistent kunne identificere en artikel, en relationsliste, et billede eller hvad som helst andet. Det bør ud fra URN alene være muligt at slå indhold op uden at man nødvendigvis kender det underliggende systemlandskab. Det vil gøre det muligt at have meget løst koblede systemer. Jo mere vi kan adskille vores systemer, jo lettere vil det være at udskifte enkelte elementer og jo lettere kan vi skalere de enkelte komponenter.

https://developer.dr.dk/docs/specs/urn.html

TODO: Mere fyld!

6.3 Underliggende infrastruktur

Alt indhold på dr.dk serves i dag igennem Akamai CDN. Akamai fungerer som Content Delivery Network, DDOS-beskyttelse og cache-lag. I tilfælde af udfald på dr.dk er cache hos Akamai sat til at blive ved med at servere nuværende data, indhtil vi igen er i stand til at besvare på forespørgsler på dr.dk. Det betyder at forlængede svartider eller korte nedbrud ikke nødvendigvis bliver bemærket hos slutbrugeren.

Infrastrukturen som i dag driver dr.dk har et par fordele men også en række ulemper. Hvert team har valgt hosting og teknologi ud fra hvad de kender til og deres lokale ønsker om funktionalitet. Det betyder at vi i dag har dele af dr.dk hostet på fem forskellige cloud platforme,

- AWS (eks. Drupal)
- Azure (dr.dk-forside, Nyhedsapp-backend)
- DR-byens datacenter (primært legacy-produkter)
- Google Cloud (produkter på DR Pipeline)
- Heroku (eks. Steffi, Hydra)

For at en artikel kan udkomme på dr.dk-forsiden og i Nyhedsapp skal der udveksles data på tværs af alle fem skyer.

Kommunikationsudfald eller nedbrud hos en af cloud-udbyderne vil i de fleste tilfælde betyde, at vi ikke kan udkomme med nye artikler. Der er ingen driftsmæssig fordel ved at vi har baseret vores infrastruktur på fem forskellige skyer, da udgivelse af en artikel vil fejle lige meget hvilket led i kæden der knækker.

6.4 Parallelliseret og distribueret system

DR har i kraft af Public Service-kontrakten med staten, en forpligtigelse til at kunne udkomme med nyheder og varslinger hele tiden og med kort varsel.

Det stiller selvfølgeligt krav til vores systemers oppetid. Jo tættere koblet et system er, jo mere sårbar er det over for nedbrud i kæden af de systemer, det er koblet til.

De systemer vi selv hoster og drifter, bør sikres i det omfang det er muligt. Jeg vil foreslå at vi kikker grundigt på et parallelt sky-setup, så vi f.eks. har en primær region (europe-west2) hvor der er veldefinerede skaleringsregler og etableret overvågning. Vi kan samtidigt have en sekundær sky sat op i en anden region (f.eks. europe-west4) som er skaleret helt i bund og ingen trafik varetager. Ved nedbrud i den primære region kan vi skifte til den sekundære og lade den skalere op.

Der er flere muligheder for hvordan et sådan setup kan fungere. Vi kan enten have et fuldt parat system allerede spundet op og parat eller vi kan sørge for at have en scriptet mulighed for at deploye et komplet system ud fra kompileret kode og database backups. Præcist hvordan dette skal sættes op og hvor meget det vil koste bliver vi nød til at kikke nærmere på når det bliver en mulighed.

6.5 DevOps-kulturen styrkes

Den primære fokus i DevOps kulturen er et forsøg på at gøre op med de forskelle der er imellem en udviklingsafdeling og en driftsafdeling. Dette er primært ud fra anerkendelsen af at udviklerne af et stykke software formentligt også er dem der bedst er i stand til at finde og rette fejl i deres software.

Et vigtigt element i dette er at det skal være muligt for udviklerne at teste i et produktionslignende miljø og at kunne deploye en rettelse til produktion med god antagelse af at det ikke har en negativ impact på resten af infrastrukturen.

Det er her relevant at kikke nærmere på elementer fra CI/CD kulturen (Continuous Integration / Continuous Deployment). Standardiseret byg pipeline, statisk kodeanalyse, afvikling af unittests og automatiseret deploy til test miljø og relateret tiltag vil være med til at finde fejl og mangler så tidligt som muligt, så releases til produktion kan ske både trygt og hurtigt.

6.5.1 Ensretning af pipeline

De pipelines der i dag benyttes på tværs af web og apps er ret optimeret i forhold til de enkelte teams og deres release process. Hvis vi går fra en specialiseret til en generel pipeline vil de enkelte teams release-process skulle tilrettes markant for blot at opnå den samme funktionalitet som i dag. Der vil dog være store fordele der kan høstes hvis pipelinen benyttes af flere teams.

- Fælles værktøjer
- Central udvikling af værktøjer
- Standardiserede værktøjer til statisk kodeanalyse
- Standardiseret deployment værktøj
- Fælles repository af deployet kode

Det vil ligeledes give den fordel at det vil være lettere for udviklerne at aflaste hinanden på tværs af teams, da værktøjer og standarder på pipelinen er fælles, så det kun er kodeforskelle der er behov for at sætte sig ind i.

6.5.2 Testmiljøer

Ideelt set skal det være muligt at deploye et komplet miljø til enten test eller produktion fra et centralt sted. Det må meget gerne være muligt at have et aktivt produktionssystem, et inaktivt produktionssystem, et aktivt test system og mulighed for deploy af dedikerede testsystemer.

- fast fuld funktionibelt testsystem baseret på seneste release (staging).
- mulighed for deploy af nyt miljø med nødvendige komponenter.

Da det aldrig vil være muligt at få vilkårligt mange testsystemer stillet til rådighed af alle vores samarbejdspartnere eller SaaS leverandører, vil vi med al sandsynlighed være tvunget til at lade flere testsystemer tilgå de samme eksterne services, hvilket normalt ikke er noget problem, men i de tilfælde hvor eksterne services har behov for at kalde tilbage til vores systemer, vil vi formentligt være nød til at etablere et proxy system, der kan route kald til de respektive testsystemer, hvis det er nødvendigt.

6.5.3 Fælles overvågning

Udviklerne i de enkelte teams er specialister i deres egne systemer og er som sådan, de bedst egnede til at fejlfinde og vedligeholde deres systemer. Det betyder dog ikke at udviklerne er de bedst egnede til at overvåge og fejlfinde i infrastrukturen som systemerne kører på. Ideelt skal vores systemer køre på en fælles platform, som der kan overvåges centralt. Ideelt set burde platformen overvåges centralt, med

veldefinerede alarmer på alle miljøer. I tilfælde af nedbrud på et miljø bør overvågningen have en drejebog på hvilke skridt der kan foretages for at afhjælpe problemerne. For at opnå det er der behov for følgende:

- Konsolidering af driftsmiljøer.
- Velbeskrevne overvågningsparametre.
- Velbeskrevet drejebog for afhjælpning af problemer.
- Velbeskrevne eskaleringsmuligheder af problemer.

Udviklerne i de enkelte teams har ansvaret for udvikling, test og vedligeholdelse af deres systemer. Udviklerne skal ligeledes sørge for at der opsættes overvågningspunkter, så der kan rapporteres systemstatus til overvågningen samt drejebøger for hvordan almindelige fejl kan afhjælpes. DevOps-afdelingen skal stå for udvikling af de nødvendige værktøjer og skal have det formelle ejerskab af infrastrukturen. Hvis der er nedbrud på netværk, hosting eller andre infrastruktur komponenter, så skal dette løses af ejeren af infrastrukturen.

Overvågningen kan varetages af Infra-afdelingen. Denne overvågning skal være af både infrastruktur og overvågningspunkter der leveres af udviklerne i de enkelte udviklingsteams. Da hverken DevOps eller de enkelte udviklingsteams har døgnbemanding, er Infra reelt eneste mulighed for at afhjælpe problemer på platformen uden for normal arbejdstid.

Opstår der et problem med infrastrukturen (udfald i tjenester hos cloud provider eller nedbrud i kubernetes setup) bør det være ejerne deraf der løser problemerne. Opstår der nedbrud i de enkelte tjenester der er deployet, som ikke kan afhjælpes ud fra den tilhørende drejebog, så bør det være de respektive udviklingsteams der har ansvaret for at afhjælpe dem.

En naturlig del af en fælles overvågning er også en fælles tilgang og standard for logning af events på tværs af platformen. Hvis vi er enige om hvordan der logges, hvad der logges og hvor meget der logges, så vil det i mange tilfælde være en lettere oplevelse at fejl finde på tværs af platformen og dermed at spotte hvor et problem opstår, ikke kun hvor et problem er tydeligt.

7 Reference Arkitektur

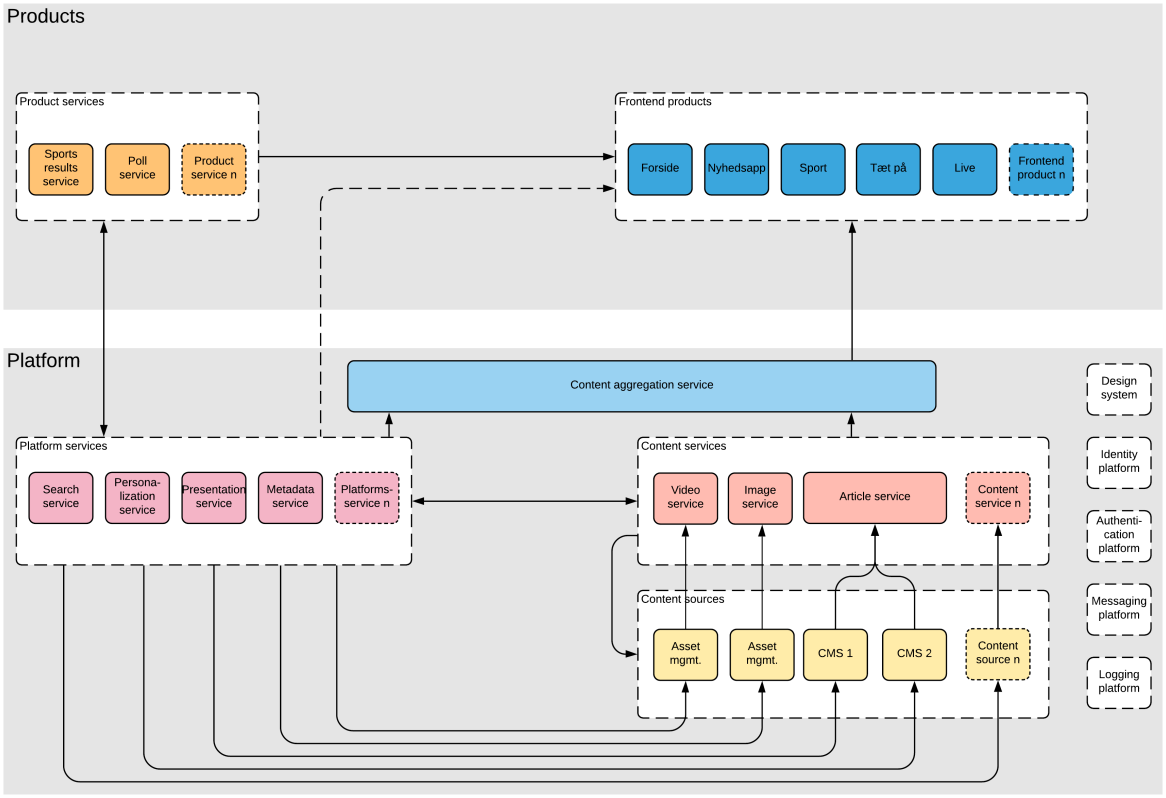

figure 1: Reference Arkitektur

7.1 Produkt / Platform adskillelse

Produkter kendetegnes ved hurtige ændringer over kortere levetid, direkte værdiudløsning overfor brugerne og højere grad af styringsmæssig autonomi. Produkternes udviklingshastighed, specialisering og tilpasningsevne kan dikteres af forretningsbehovet så frem der tages hensyn til platformskomponenterne der skal anvendes.

Platformen kendetegnes ved færre ændringer over lang levetid, indirekte værdiudløsning gennem produkter og styringsmæssig alignment på tværs af tid og produkter. Platformens robusthed og tværgående værdi skal ikke kompromitteres af produkternes specifikke implementeringer, lokale beslutninger eller korte levetid. En platformskomponent kan ikke afhænge af en produktkomponent.

- Produkter udvikles ovenpå og anvender platformens komponenter og opfylder specifikke produktkrav som platformen ikke selv kan eller skal.
- Produkter skal have et meget snævert fokus, kort levetid og kan endda udvikles ud fra stik modsatte principper end platformen.
- Produkter kan let tilpasse sig skiftende kundespecifikke behov.
- Produkters lifecycle skal forventningsafstemmes inden etablering, og løbende justeres om nødvendigt.
- Produkt komponenter kan forfremmes til platform komponenter hvis de overholder den for platformen gældende governance. Produkt komponenter bør forfremmes hvis en produkt komponent skal anvendes af flere forskellige produkter.
- Platform komponenter har ofte en lang levetid og må forvente at der er mange aftagere. Når en platformskomponent ændres bør der tages højde for de mange aftagere.

- Platformen kan implementere ændringer og understøtte funktionalitet, hvis værdiudløsning bevidst først kommer efter adskillige iterationer og et større antal produkter.
- Platformen har en stram governance, hvor ændringer skal imødekomme et større antal aftagere, således at ændringer ikke afføder uventede konsekvenser hos platformens aftagere.

Denne opdeling passer godt med Gartners PACE layered application strategy, der forsøger at give et fornuftigt bud på application lifecycle management.

https://cio-wiki.org/wiki/Gartner%27s_PACE_Layered_Application_Strategy

7.2 Systemgruppering

I det følgende beskrives de forskellige grupperinger i Referencearkitekturen kort.

7.2.1 Content Sources

Content Sources er de indholds bærende systemer. Dette kan f.eks. være Drupal, DR Billeder eller MU-Online. Content Sources indeholder oftest data der udstilles mange forskellige steder. Data fra Content Sources udstilles udelukkende igennem Content Services, men Content Sources kan opdateres fra andre fra andre systemer der kalder Content Sources direkte.

7.2.2 Content Services

Content Services er de services der som oftest er bygget oven på de enkelte Content Sources. Content Services benyttes oftest til at skærme Content Sources fra at skulle overholde kontrakter over for mange forskellige aftagere og for at beskytte mod tunge belastninger af Content Sources. Eksempler på Content Services kan f.eks. være Mimer 4 eller DR Billeder.

7.2.3 Platform Services

Platform Services er centralle services der udstiller funktioner og data der benyttes af flere aftagere. Eksempler på Platform Services kan være Image Scaler, Midas eller Garnnøgle.

7.2.4 Product Services

7.2.5 Frontend Products

8 Analyse af komponenter

8.1 Akamai

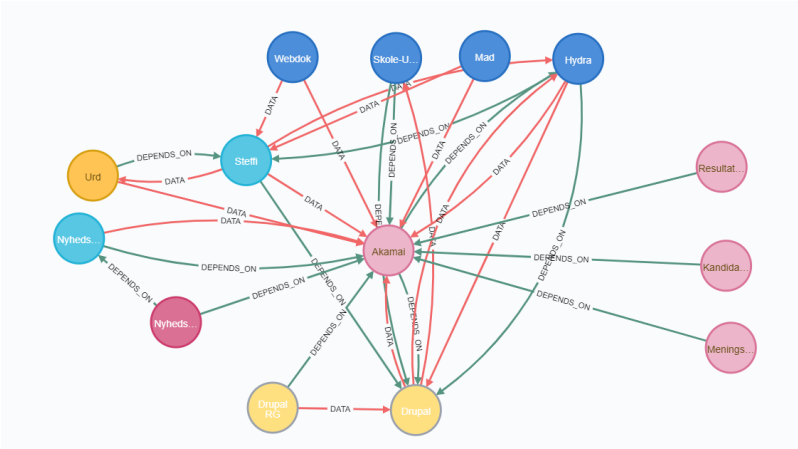

figure 2: `MATCH (x{name:'Akamai'})--(y) RETURN x,y`

Kort beskrivelse

Akamai benyttes som det primære CDN (Content delivery Network). Det er dermed ikke et produkt som er udviklet hverken af eller for DR, men et produkt som vi er afhængige af.

Anbefalet handling

Akamai som CDN udfylder fint de behov som DR har. Vi kan med fordel kikke på om vi benytter fornuftige cache timeouts eller om vores hjemmeside er struktureret til at få mest muligt ud af CDN.

Overslag

Indgår ikke som sådan i roadmap.

8.2 Hydra

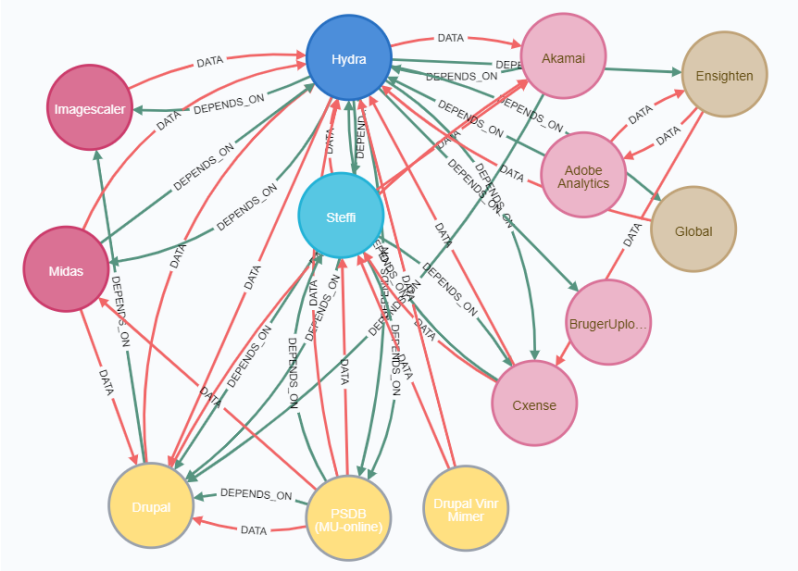

figure 3: `MATCH (x{name:'Hydra'})--(y) RETURN x,y`

Kort beskrivelse

Web-frontend, der varetager indholdsvisning for artikler og forsider. Anvender Drupal-visning til sider, der endnu ikke understøttes af Hydra. Ovenstående figur viser at der er et højt antal afhængigheder til mange systemer. Det skal undersøges nærmere om de alle er aktuelle.

Anbefalet handling

Hydra er beskrevet som at den direkte henter data fra og sender data til Drupal. Er det korrekt, så springer den flere lag over i referencearkitekturen. Hydra burde kun afhænge af platform utilities, platform services, product services eller content aggregation.

8.3 Steffi

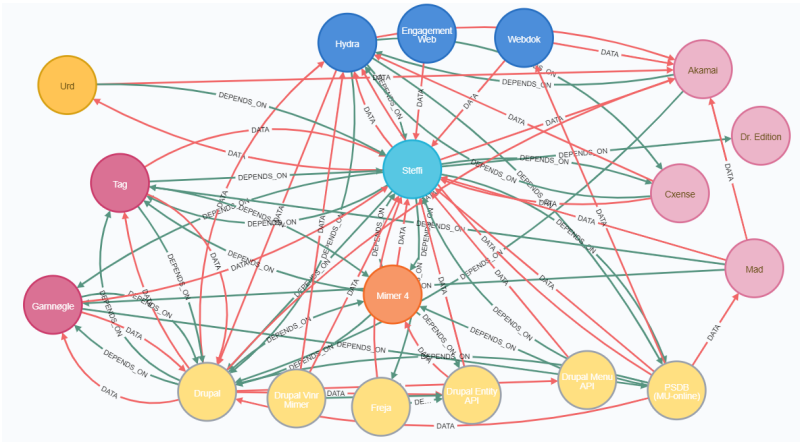

figure 4: `MATCH (x{name:'Steffi'})--(y) RETURN x,y`

Kort beskrivelse

Forespørgselslag, der tilbyder GraphQL-grænseflader til frontendapplikationer. Benytter Content aggregering. Der findes to kopier af Steffie i produktion, en der benytter en kort cache til hyppigt adspurgte ressourcer og en der ikke har nogen cache. Der er ingen cache invaliderings strategier eller mulighed for at reagere på events fra underliggende systemer.

Anbefalet handling

Steffie er vores akutelle bud på et content aggregation lag.

8.4 Talentholdet

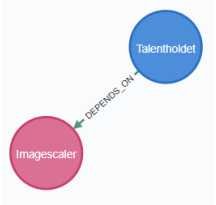

figure 5: `MATCH (x{name:'Talentholdet'})--(y) RETURN x,y`

Kort beskrivelse

Talentholdet er en rekrutteringsplatform for nye talenter til DR. Talentholdet har sin egen database hvori den gemmer personidentificerbar data i krypteret form. Talentholdet afhænger af Imagescaler

Anbefalet handling

Talentholdet udfører en mindre opgave som det er muligt at diskutere værdien af. Vi bør få afklaret med forretningen om vi eventuelt kan pensionere programmet. Der er en del kodegæld i programmet og GDPR-afledte sletninger er en manuel process. Prioriteten af Talentholdet er dog ret lav, så den manuelle process kan tolereres.

8.5 Elements

figure 6: `MATCH (x{name:'Elements'})--(y) RETURN x,y`

Kort beskrivelse

Elements er DR's centrale komponentbibliotek, som består af frontendkomponenter. Elements er kodet i React. Elements benyttes af en række af vores frontend-produkter, men ikke alle.

Anbefalet handling

Elements benyttes af en række af vores frontend-produkter og passer fint ind i vores referencearkitektur.

8.6 Webdok

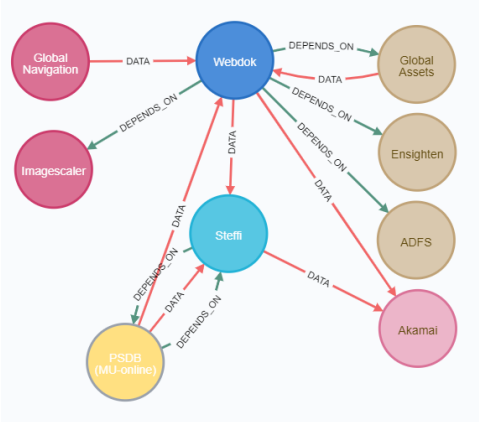

figure 7: `MATCH (x{name:'Webdok'})--(y) RETURN x,y`

Kort beskrivelse

CMS og præsentation af featureartikler i særformater. Præsentationslag (Node.js) trækker data fra API (Node.js). Redaktørgrænseflade er bygget ind i præsentationslaget.

Anbefalet handling

Hvordan Webdok er flettet ind i vores nuværende arkitektur skal undersøges nærmere. Dokumentationen som den står nu giver et lidt mudret billede

8.7 Skole-Undervisning

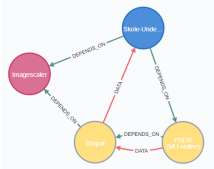

figure 8: `MATCH (x{name:'Skole-Undervisning'})--(y) RETURN x,y`

Kort beskrivelse

Skole og Undervisning er et site, der formidler undervisningsforløb og undervisningsmateriale til Skoler. Sitet er opbygget i Drupal, men har en række specialløsninger konstrueret for at kunne samle temaer og autogenerere faktabokse etc.

Anbefalet handling

Skole-Undervisning passer ikke ind i referencearkitekturen, idet den udgør sin helt egen silo og udstiller data direkte til slutbrugere via Akamai. Afhængigt af levetiden for projektet (Gartners TIME-model) skal vi overveje hvad vi gør ved projektet.

8.8 Mad

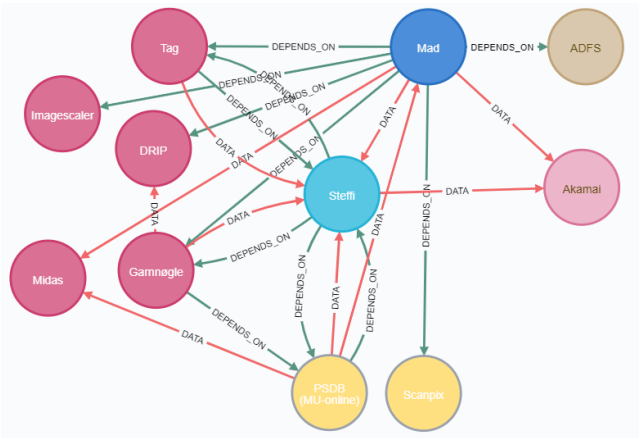

figure 9: `MATCH (x{name:'Mad'})--(y) RETURN x,y`

Kort beskrivelse

Mad er en applikation, der håndterer opskrifter, artikler og opskriftssamlinger. Mad er bygget som Headless Drupal med egen RG til opskrifter. Artikler publiceres gennem Drupal.

Anbefalet handling

Mad passer ikke helt ind i referencearkitekturen som beskrevet. Om det er en mangel i dokumentationen eller i arkitekturen skal undersøges. (hvorfor tilgår Steffi Mad direkte og ikke igennem f.eks. Mimer?)

8.9 Nyhedsapp-Frontend

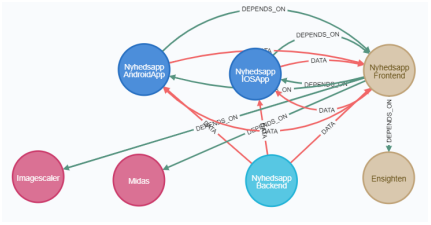

figure 10: `MATCH (x{name:'Nyhedsapp Frontend'})--(y) RETURN x,y`

Kort beskrivelse

Præsentationslag for indhold i Nyhedsapp, således at visningen mellem iOS og Android er homogen, samt har lignende homogenitet til visningen på DR.dk Nyhedsapp-Frontend er ikke en selvstændig applikation, men et delt kodelag der anvendes af iOS- og Android-udgaverne af Nyhedsapp.

Anbefalet handling

Nyhedsapp-Frontend eksisterer kun i kraft af den aktuelle implementering af Nyhedsappen. Hvis Nyhedsappen skal opdateres eller udskiftes vil Nyhedsapp-Frontend også skulle opdateres eller skiftes.

8.10 Nyhedsapp-iOS

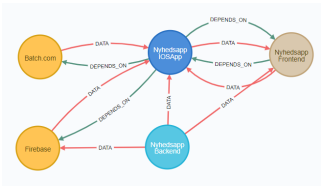

figure 11: `MATCH (x{name:'Nyhedsapp IOSApp'})--(y) RETURN x,y`

Kort beskrivelse

Afvikler Nyhedsapp-frontend, integrerer til iOS-native funktionalitet

Anbefalet handling

Nyhedsapp til både iOS og Android overholder referencearkitekturen. Der er måske ønsker om at opdatere applikationerne, så de får et mere moderne udtryk på telefonerne. Dette ønske er dog et forretningsdrevet ønske og ikke en nødvendighed set ud fra referencearkitekturen. Dette forretningsønske kan dog meget vel være med til at drive en række af de ændringer der vil være nødvendige for at programstakken under nyhedsappen bliver mere i tråd med referencearkitekturen.

8.11 Nyhedsapp-Android

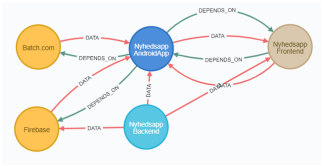

figure 12: `MATCH (x{name:'Nyhedsapp AndroidApp'})--(y) RETURN x,y`

Kort beskrivelse

Native Android-wrapper, der afvikler Nyhedsapp-Frontend

Anbefalet handling

Nyhedsapp til både iOS og Android overholder referencearkitekturen. Der er måske ønsker om at opdatere applikationerne så de får et mere moderne udtryk på telefonerne. Dette ønske er dog et forretningsdrevet ønske og ikke en nødvendighed set ud fra referencearkitekturen. Dette forretningsønske kan dog meget vel være med til at drive en række af de ændringer der vil være nødvendige for at programstakken under nyhedsappen bliver mere i tråd med referencearkitekturen.

8.12 Batch.com

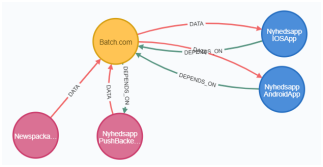

figure 13: `MATCH (x{name:'Batch.com'})--(y) RETURN x,y`

Kort beskrivelse

SaaS tjeneste der benyttes til push af beskeder til Nyhedsapp. Distribuerer data til Nyhedsapp-instanser, med mulighed for dublikeringskontrol så den samme enhed kun modtager data én gang, og ligeledes en opfølgningsmulighed så devices der var utilgængelige på udsendelsestidspunktet, opdateres når de er tilgængelige igen

Anbefalet handling

Tjenesten passer ind i vores referencearkitektur. Om Batch.com bliver ved med at være den bedste løsning for push af beskeder vil vi ikke tage stilling til i dette dokument.

8.13 Firebase

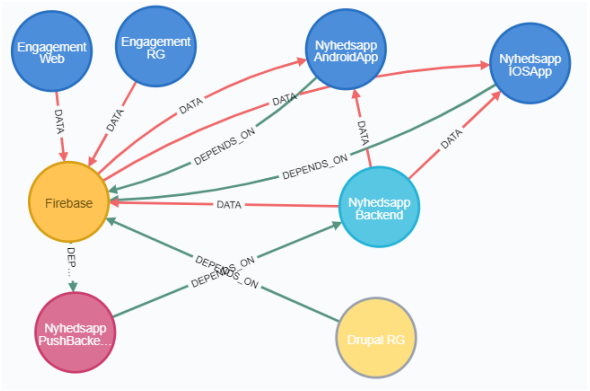

figure 14: `MATCH (x{name:'Firebase'})--(y) RETURN x,y`

Kort beskrivelse

SaaS tjeneste der benyttes til Push-service af flere applikationer: Nyhedsapp, Drupal RG ("hvem er inde på min artikel")

Anbefalet handling

Tjenesten passer ind i vores referencearkitektur. Om Firebase bliver ved med at være den bedste løsning for push af beskeder vil vi ikke tage stilling til i dette dokument.

8.14 OCS

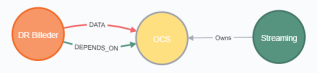

figure 15: `MATCH (x{name:'OCS'})--(y) RETURN x,y`

Kort beskrivelse

API-afløseren til PSDB (MU-online), der leverer video- og radio-data (ikke anvendt i Web & Apps endnu).

Udstiller program- og seriedata, således at aftagere kan benytte dette til at vise TV/Radio-indhold

Anbefalet handling

Ved ændringer i systemer der afhænger af PSDB, så bør det undersøges om det er relevant at anvende OCS i stedet.

8.15 Garnnøgle

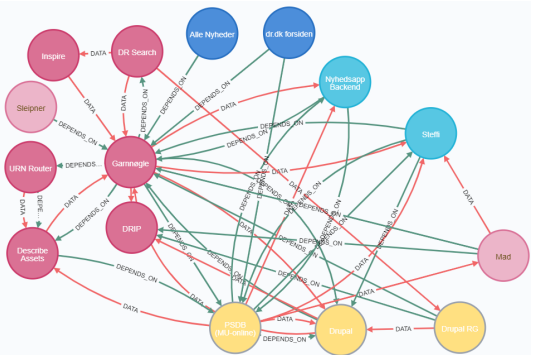

figure 16: `MATCH (x{name:'Garnnøgle'})--(y) RETURN x,y`

Kort beskrivelse

Tema- og sagskategorisering af artikelindhold, således specielt nyhedsindhold kan inddeles i overordnede kategorier. Bruges også som emnebaseret abonnement-mekanisme i Nyhedsapp Garnnøgle er ret central i værdikæden for DR.dk, da den benyttes af en lang række systemer og funktioner. Garnnøgle applikationen er behæftet med en del teknisk gæld og unødig kompleksitet i dens implementering.

Anbefalet handling

Garnnøgles placering i forhold til referencearkitekturen er ok. Selve implementeringen af garnnøgle bør dog genbesøges og afhængigt af det reelle forretningsbehov skal vi have set på en ny implementering.

8.16 Drupal

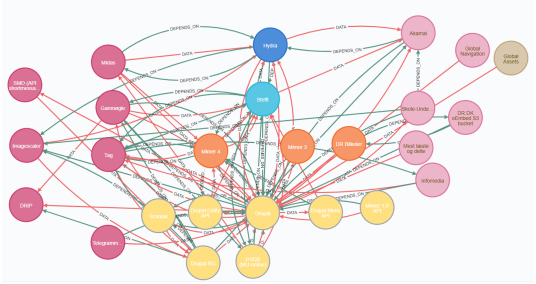

figure 17: `MATCH (x{name:'Drupal'})--(y) RETURN x,y`

Kort beskrivelse

Det primære CMS til tekstbaseret indhold på DR.dk, samt redaktionel grænseflade til indholdsproduktion og slutteligt mulighed for at grafisk opbygge sektionsforsider og sites.

Anbefalet handling

I henhold til referencearkitekturen burde Drupal kun være forbundet med Content Services og Platform Services. Da Drupal er det centrale CMS system er antallet af forbindelser ikke den store overraskelse, det peger dog entydigt på at DR stadig er meget hængt op på Drupal og dermed vil en eventuel udskiftning eller opdatering af Drupal være en stor udgift. Vi bør helt klart se på at nedbringe antallet af direkte afhængigheder til og fra Drupal. Dette kan enten ske ved at lade Platform Services snakke med Content Services (f.eks. at lade garnnøgle afhænge af Mimer i stedet for Drupal), affolke og lukke de gamle udgaver af Mimer eller sikre at kun services med en god grund kan kommunikere med Drupal.

Drupal er lige nu vores styrende CMS system, det vil sige at Drupal har kontrollen med alle URL hos DR.dk, hvilket også er grunden til den direkte forbindelse fra Hydra ned til Drupal. Det vil være meget fornuftigt at kikke på at få etableret et fornuftigt URN / URL system, således at en URL / URN i sig selv er beskrivende nok til at kunne udpege det styrende indholdssystem. Det vil muliggøre sideløbende CMS systemer med hvert deres fokus og gøre det lettere at opdatere / udskifte Drupal på sigt.

Vi skal også have kikket på roadmap for opdatering af Drupal 7 til enten Drupal 9 eller et andet CMS system. Drupal 7 (vores nuværende installation) og efterfølgeren Drupal 8 har end of life i november 2021.

Overslag

Migrering fra Drupal 7 til andet CMS system vil være en betydelig opgave, især med det niveau af afhængigheder og egenudvikling der er foretaget på Drupal platformen.

8.17 Mimer 1.0 API

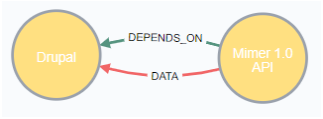

figure 18: `MATCH (x{name:'Mimer 1.0 API'})--(y) RETURN x,y`

Kort beskrivelse

Mimer 1.0 API er nogle REST services i Drupal, som giver Mimer 1.1 adgang til at oprette nyt indhold.

Anbefalet handling

Mimer 1.0 API er allerede under afvikling. Denne afvikling bør fortsættes og systemet lukkes.

8.18 Drupal RG

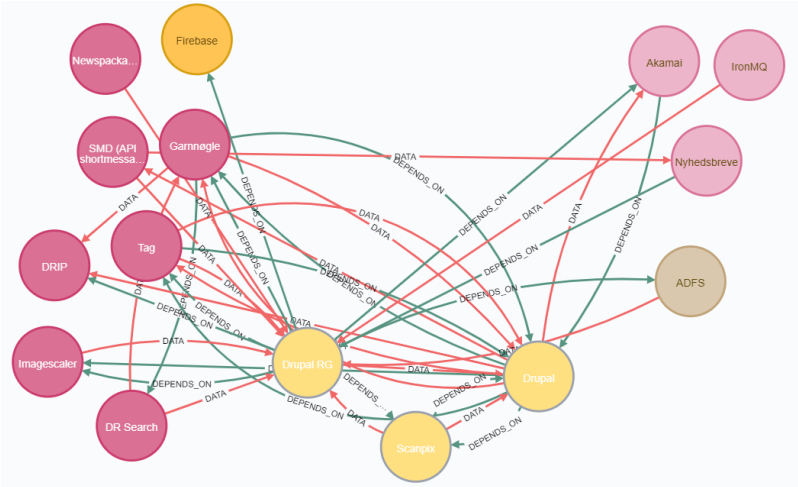

figure 19: `MATCH (x{name:'Drupal RG'})--(y) RETURN x,y`

Kort beskrivelse

Nyeste version af redaktør grænsefladen, lanceret Q1-2018. Redaktør grænsefladen (RG) er journalisternes værktøj til artikelproduktion. Den findes i to udgaver, hhv. RG1 og RG2.

Webbaseret grænseflade til at oprette, redigere og distribuere tekstbaseret indhold, primært artikler. Tilbyder også integration med tema, kategoriserings, og push-funktioner.

8.19 Drupal Entity API

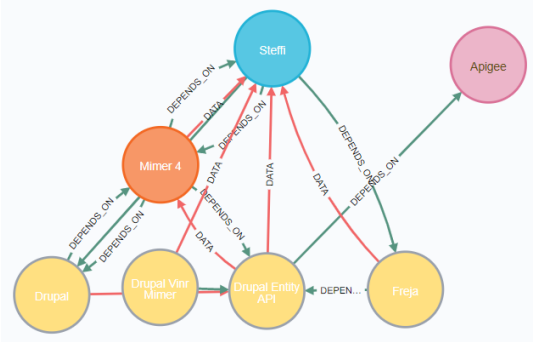

figure 20: `MATCH (x{name:'Drupal Entity API'})--(y) RETURN x,y`

Kort beskrivelse

Udstiller væsentlige dele af Drupals datamodel for indholdstyper i et simpelt API

8.20 Drupal Menu API

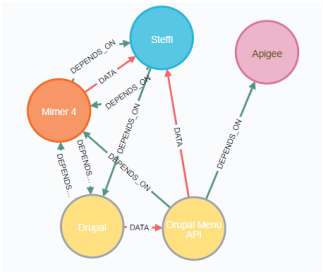

figure 21: `MATCH (x{name:'Drupal Menu API'})--(y) RETURN x,y`

Kort beskrivelse

Udstiller Drupals menu og sitestruktur til aftagere, der ønsker at præsentere indhold i en hierarkisk visning.

8.21 Ensighten

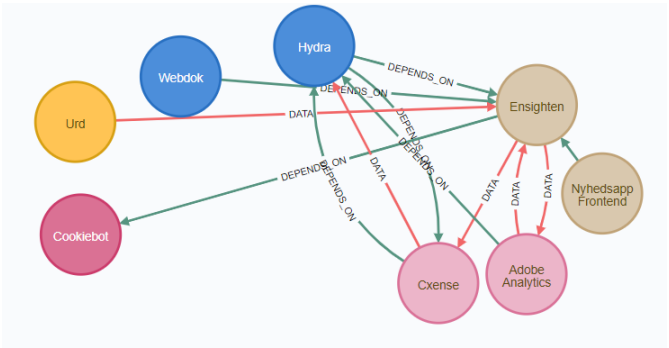

figure 22: `MATCH (x{name:'Ensighten'})--(y) RETURN x,y`

Kort beskrivelse

Indskyder JavaScript på dr.dk sider, f.eks. til analytics, bannere, mv. Sørger derudover også for at hente og behandle metadata til analytics og anbefalingssystemer.

8.22 Ad Server

Kort beskrivelse

"Smart Ad Server" er en hosted løsning, hvor bannerhåndtering og booking/kampagner bliver håndteret. Integreres med DR applikationer der skal vise bannere til slutbrugere.

Anbefalet handling

SaaS. Der er ingen udviklingsmuligheder her, kun eventuelt at fjerne afhængigheden heraf hvis nødvendigt.

8.23 Midas

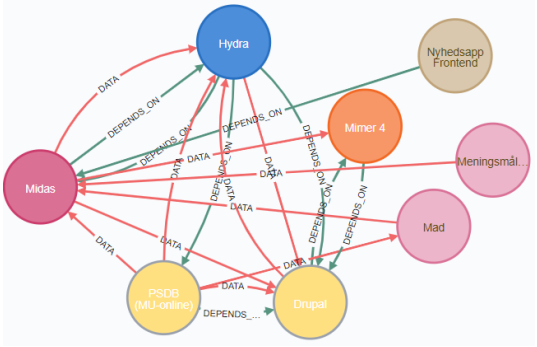

figure 23: `MATCH (x{name:'Midas'})--(y) RETURN x,y`

Kort beskrivelse

Oembed-service, der omsætter oEmbed-referencer til markup, til brug i frontend.

8.24 PSDB (MU-online)

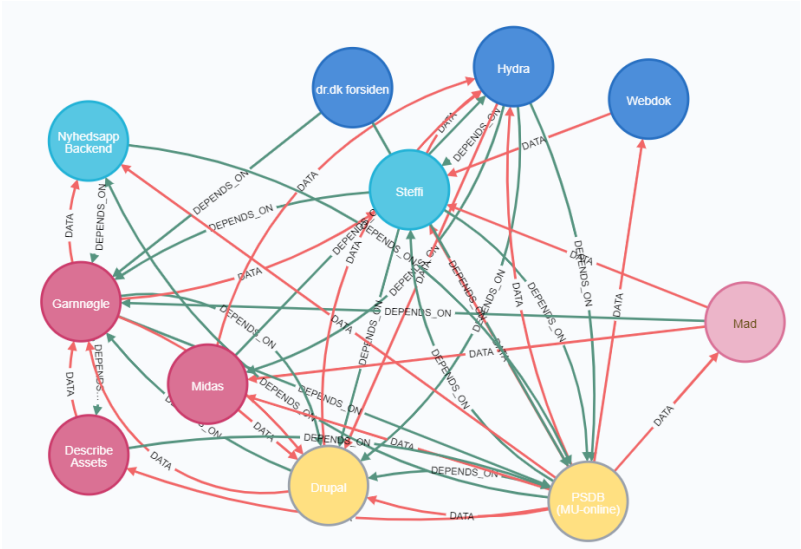

figure 24: `MATCH (x{name:'PSDB (MU-online)'})--(y) RETURN x,y`

Kort beskrivelse

Program og seriedatabase til TV-baseret indhold

Anbefalet handling

Ejerskabet af dette system ligger uden for Web og Apps.

8.25 Scanpix

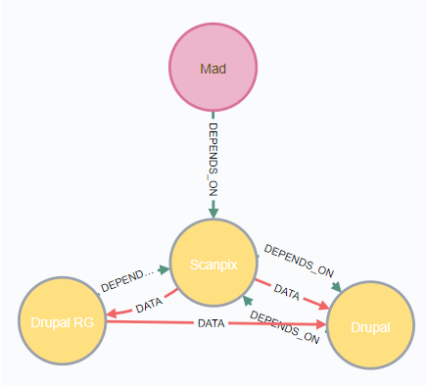

figure 25: `MATCH (x{name:'Scanpix'})--(y) RETURN x,y`

Kort beskrivelse

Billededatabase. Ekstern service, der tilbyder redaktører at søge efter billed- og videoindhold som kan importeres og anvendes i DR's systemer.

Anbefalet handling

SaaS. Der er ingen udviklingsmuligheder her, kun eventuelt at fjerne afhængigheden heraf hvis nødvendigt.

Overslag

8.26 Cxense

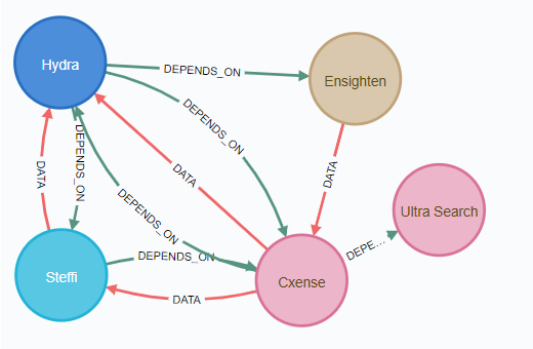

figure 26: `MATCH (x{name:'Cxense'})--(y) RETURN x,y`

Kort beskrivelse

Analysesystem til personalisering

Anbefalet handling

SaaS. Der er ingen udviklingsmuligheder her, kun eventuelt at fjerne afhængigheden heraf hvis nødvendigt.

8.27 Adobe Analytics

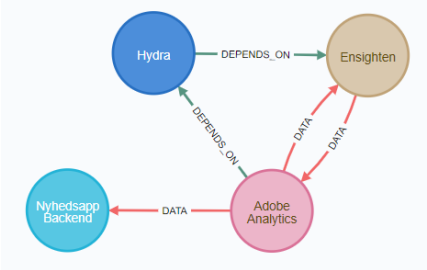

figure 27: `MATCH (x{name:'Adobe Analytics'})--(y) RETURN x,y`

Kort beskrivelse

Analysesystem til personalisering

Anbefalet handling

SaaS. Der er ingen udviklingsmuligheder her, kun eventuelt at fjerne afhængigheden heraf hvis nødvendigt.

8.28 Pressebilleder

Kort beskrivelse

Pressesite på dr.dk Udsendelse af nyhedsbreve, integration til Mosaic-billeddatabase. Sandsynligvis eksternt produkt.

8.29 WebCMS

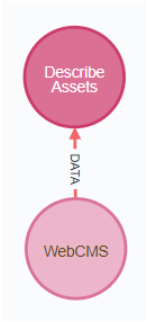

figure 28: `MATCH (x{name:'WebCMS'})--(y) RETURN x,y`

Kort beskrivelse

Content Management System håndtering af sider på dr.dk. Udfaset og afløst af Drupal.

8.30 DR Search

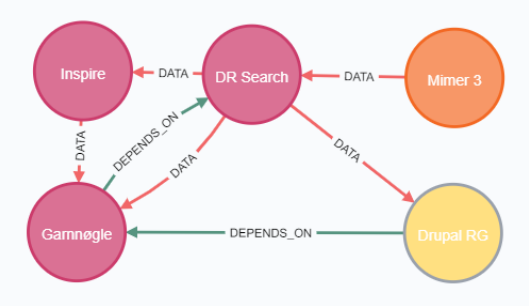

figure 29: `MATCH (x{name:'DR Search'})--(y) RETURN x,y`

Kort beskrivelse

Søgemaskine tekst, video og radioindhold på DR.dk

Tilbyder søgefunktionalitet med ekstrem domæneforståelse for DR's forskellige indholdstyper, distributionsregler og indholdssystemer. Kan indlejres i RG-systemer og tilbyde specialiseret søgefunktionalitet til både publiceret og ikke-publiceret indhold. Kan tilbyde slutbrugere at søge agnostisk og præsenterer søgeresultater ud fra den type indhold der er tale om, eks afspilningsoplysninger til TV-indhold. Indekserer de forskellige indholdssystemer for nyt eller opdateret indhold, og indekserer også indhold via webvisning på DR.dk

Anbefalet handling

DR Search er for tiden uden ejerskab og uden ansvarlige udviklere. Enten skal systemet adopteres af at team eller også så skal vi have afviklet systemet. Der er ikke umiddelbart et system der kan tage over som erstatning for DR Search endnu.

* Skal vi erstatte? * Skal vi beholde? * Skal det helt skrottes? * Kan vi benytte Drupals search til artiker, og evt streamingtjenestens søgefunktionalitet til TV / Radio?

Overslag

8.31 Nyhedsbreve

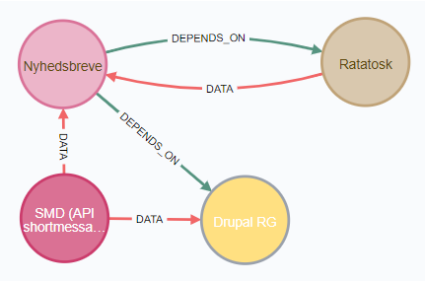

figure 30: `MATCH (x{name:'Nyhedsbreve'})--(y) RETURN x,y`

Kort beskrivelse

Udsendelse af nyhedsbreve. Benytter Peytz' service

Tilbyder nyhedsbreve som en distributionskanal til DR's indholdsysstemer, således at indhold kan skubbes ud til brugeren samtidig med at det publiceres på DR.dk

Anbefalet handling

SaaS men uden ejerskab. Det skal vurderes om denne løsning er passende for vores fremtidige arkitektur. Det er dog ikke højeste prioritet.

8.32 DR Billeder

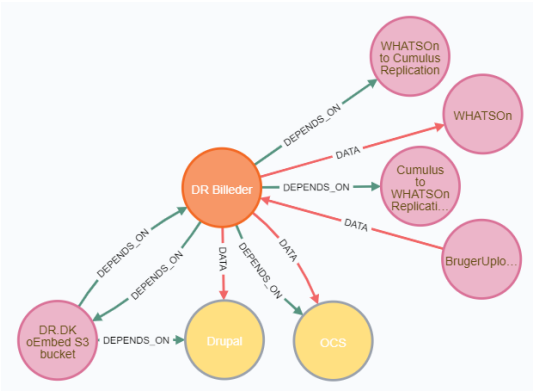

figure 31: `MATCH (x{name:'DR Billeder'})--(y) RETURN x,y`

Kort beskrivelse

Billedhåndteringssystem til DR's indholdsproducenter, anvender standardsystemet Cumulus fra Canto. Der anvendes en underleverandør, Attention, til at konfigurere systemet. Består af følgende delkomponenter: Database, applikationsserver, web UI, tyk klient.

Anbefalet handling

Ingen handling nødvendig. Systemet håndteres uden for Web og Apps

8.33 DR.DK oEmbed S3 bucket

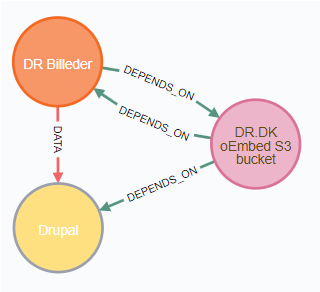

figure 32: `MATCH (x{name:'DR.DK oEmbed S3 bucket'})--(y) RETURN x,y`

Kort beskrivelse

En AWS S3 bucket der anvendes til online versioner af billeder som indlejres med oEmbed

Anbefalet handling

Ingen handling nødvendig. Systemet håndteres uden for Web og Apps

8.34 Mimer 3

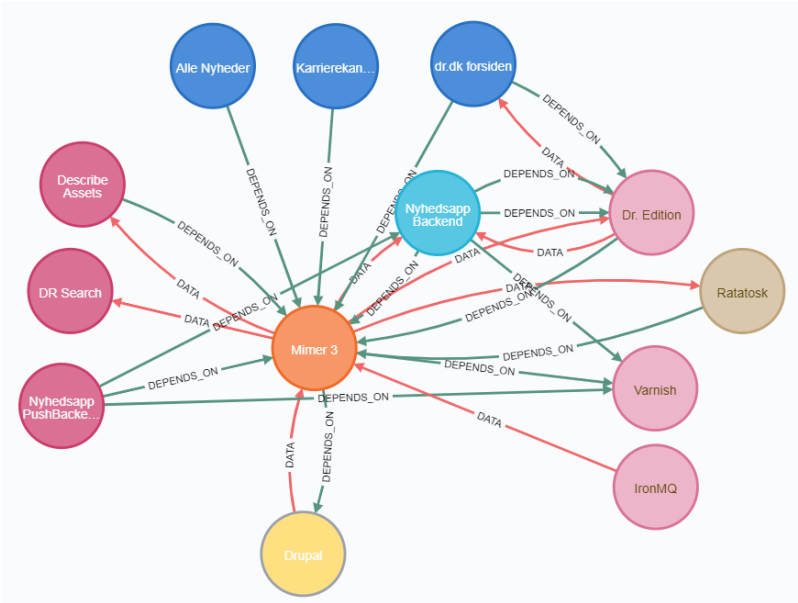

figure 33: `MATCH (x{name:'Mimer 3'})--(y) RETURN x,y`

Kort beskrivelse

API til Drupals artikelindhold.

Udstiller Drupals primære indholdstyper til brug og visning i andre systemer.

Anbefalet handling

Mimer 3 er under afvikling. Det viste billede af afhængigheder er stadig ved at skrumpe. Mimer 3 bør afvikles helt og fjernes, så vi slipper for at have parallelle implementeringer der løser de samme opgaver.

Overslag

Der mangler stadig mange afhængigheder der skal migreres over på Mimer 4. Disse er dog i pipeline.

8.35 Mimer 4

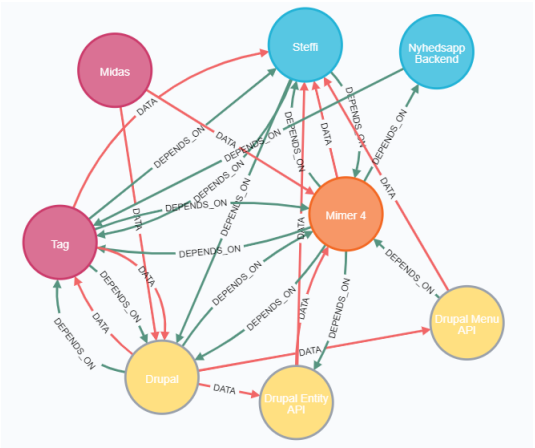

figure 34: `MATCH (x{name:'Mimer 4'})--(y) RETURN x,y`

Kort beskrivelse

API til Drupals artikelindhold.

Udstiller Drupals primære indholdstyper til brug og visning i andre systemer.

Anbefalet handling

8.36 Tag

figure 35: `MATCH (x{name:'Tag'})--(y) RETURN x,y`

Kort beskrivelse

Tagsystem til opmærkning af DR's web-indhold.

Systemet bliver ikke længere kaldt eller resultater heraf vist på dr.dk

Anbefalet handling

Tag manager systemet bliver slet ikke brugt i det omfang det burde. Sidste opdatering er fra marts 2018. Vi bør sammen med opdateringen af garnnøgle se på om vi kan enten bygge Tag systemet sammen med Garnnøgles erstatning eller helt udskifte Tag systemet til noget der stemmer mere overens med de behov vi har.

8.37 Urd

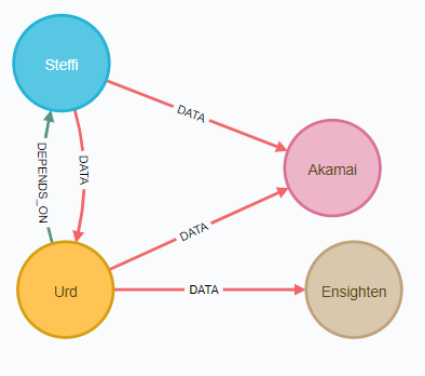

figure 36: `MATCH (x{name:'Urd'})--(y) RETURN x,y`

Kort beskrivelse

Metadataservice, der leverer metadata til brug i TMS

Anbefalet handling

8.38 Infomedia

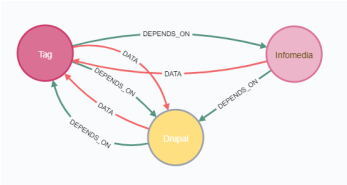

figure 37: `MATCH (x{name:'Infomedia'})--(y) RETURN x,y`

Kort beskrivelse

Taksonomidatabase og tekstanalysesystem til tags.

Tilbyder en komplet, vedligeholdt taksonomi samt tekstanalysefunktionalitet, der kan analysere en given tekst og indikere tags fra taksonomien som er relevante.

Systemet bliver ikke længere kaldt eller resultater heraf vist på dr.dk

Anbefalet handling

Infomedia bruges idag til tag systemet. Tag systemet og Garnnøgle står formentligt over for en større omskrivning. Om Infomedia skal benyttes til tekstanalyse og forslag til Tagging af artikler skal vi have set nærmere på.

8.39 Auth0

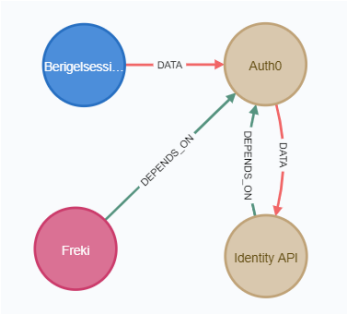

figure 38: `MATCH (x{name:'Auth0'})--(y) RETURN x,y`

Kort beskrivelse

Oauth autentifikations platform. SaaS.

Anbefalet handling

Auth0 er en fin platform til autentificering af brugere. Det er en betalt service, der passer godt ind i vores eksisterende programstak.

8.40 Identity API

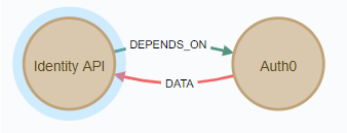

figure 39: `MATCH (x{name:'Identity API'})--(y) RETURN x,y`

Kort beskrivelse

Backend til DR's login-løsning.

Anbefalet handling

Ingen nødvendig

8.41 Telegrammaskine

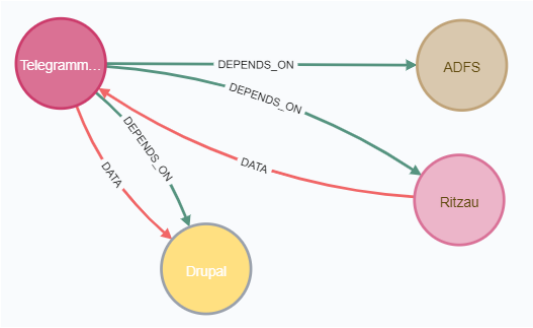

figure 40: `MATCH (x{name:'Telegrammaskine'})--(y) RETURN x,y`

Kort beskrivelse

Integration til Ritzau-indhold, med mulighed for at redigere og udgive telegrammer som Drupal-artikler.

DR modtager løbende nyhedsunderretninger fra Ritzau i form af telegrammer; Telegrammaskinen tilbyder redaktører en meget hurtig måde at overskue seneste telegrammer samt at publicere disse direkte på DR.dk

Anbefalet handling

Ingen ændringer nødvendigt.

8.42 Dr. Edition

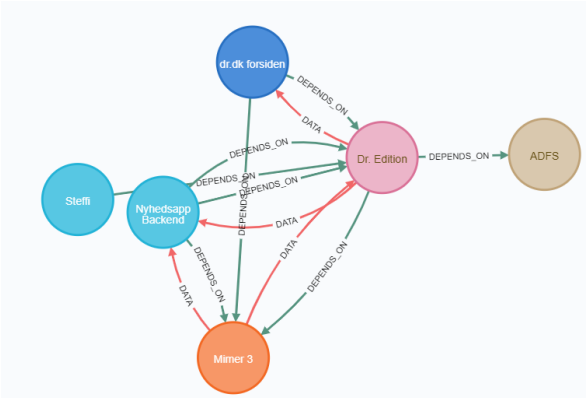

figure 41: `MATCH (x{name:'Dr. Edition'})--(y) RETURN x,y`

Kort beskrivelse

Forsideværktøj til forsiden af DR.dk, hvor redaktører udarbejder forsider, inkluderer også API. Benytter Aptomas Dr. Edition-produkt. (steffi kommer snart til at trække til den nye forside).

Tilbyder forsideredaktører at opbygge og redigere forsiden af DR.dk ud fra seneste udgivne artikelindhold fra Drupal, samt at versionere og skifte mellem flere forskellige forsideudgaver.

Anbefalet handling

8.43 dr.dk forsiden

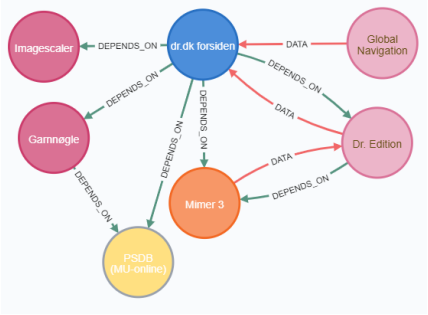

figure 42: `MATCH (x{name:'dr.dk forsiden'})--(y) RETURN x,y`

Kort beskrivelse

Azure-driftet ASP.NET MVC applikation til at hente data og vise forsiden af dr.dk

Robust system til visning af de forsider som forsideredaktører udarbejder i Dr. Edition.

Anbefalet handling

8.44 Alle Nyheder

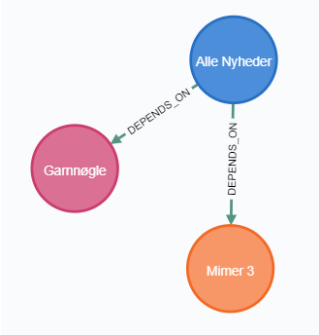

figure 43: `MATCH (x{name:'Alle Nyheder'})--(y) RETURN x,y`

Kort beskrivelse

Systemet der driver siden allenyheder (alle nyheder): http://www.dr.dk/nyheder/allenyheder/

Tilbyder overblik over senest publiceret artikelindhold samt en søgning ud fra publiceringstidspunkt

/allenyheder/ bliver ikke længere kaldt og er dekommisioneret.

Anbefalet handling

8.45 Newsapp UI

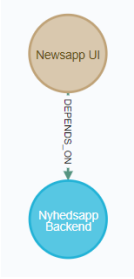

figure 44: `MATCH (x{name:'Newsapp UI'})--(y) RETURN x,y`

Kort beskrivelse

Web-renderingsværktøj til DR Nyheder (app). Web-hostet værktøj til at se mobil app i web-version. Placeret på https://www.dr.dk/tjenester/newsapp-ui/.

Redaktørvendt visning af hvordan indhold vises i Nyhedsapp, således at redaktører kan lave Nyhedsapp-specifikt preview af indhold.

Anbefalet handling

Ingen nødvendig. Applikationen lever eller dør med de 2 nyheds App

8.46 SMD (API shortmessage-dispatcher)

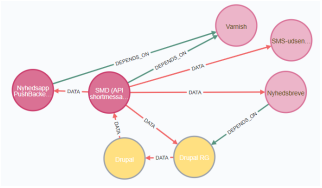

figure 45: `MATCH (x{name:'SMD (API shortmessage-dispatcher)'})--(y) RETURN x,y`

Kort beskrivelse

Push SMS; nyhedsbrev, Nyheds app. fra f.eks. Drupal, facebook.

Tilbyder push af tekstbaseret indhold til en række forskellige platforme og distributionsmekanismer

Anbefalet handling

8.47 Mest læste og delte

figure 46: `MATCH (x{name:'Mest læste og delte'})--(y) RETURN x,y`

Kort beskrivelse

ESI mest læste og delte mm. på DR.dk og Drupal sider.

Tilbyder en liste af mest besøgt og delt (via social medier) indhold ud fra besøgsstatistik.

Anbefalet handling

Applikationen skal lukkes ned, da dens funktionalitet bliver erstattet af Cxense.

8.48 Nyhedsapp Backend

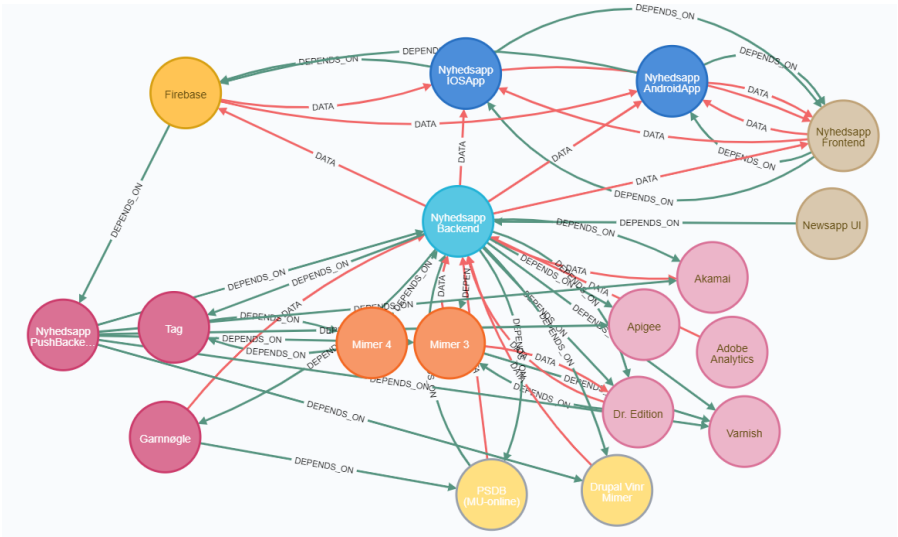

figure 47: `MATCH (x{name:'Nyhedsapp Backend'})--(y) RETURN x,y`

Kort beskrivelse

Backend til DR's Nyhedsapp.

Applikation der indsamler, aggregerer og transformerer artikler til DRs nyhedsapps dataformat. Persisterer og udstiller data for Nyhedsapp (Nyhedsapp Frontend).

Anbefalet handling

Nyhedsapp Backend forekommer også som en parallel implementering af funktionalitet der allerede eksisterer andet steds i DR.dk. Artikeldata burde blive udstillet igennem Steffi på lige fod med hjemmesiden og Nyhedsapp backend burde kun varetage funktionaliteten der adskiller Nyhedsapp fra hjemmesiden. Det vil sige push beskeder, brugerstyrring, abonnementer og lignende.

Overslag

Det virker ikke rationelt at omskrive nyhedsapp kun med henblik på at rydde lidt op i systemlandskabet, men når vi på et tidspunkt skal genbesøge nyhedsapp med henblik på modernisering bør vi også sikre at vi følger den anbefalede arkitektur og dermed indhenter content i veldefineret format fra den fælles content aggregation service.

8.49 Tag Admin

figure 48: `MATCH (x{name:'Tag Admin'})--(y) RETURN x,y`

Kort beskrivelse

Administrative interface for Tag API. Rent frontend produkt der er integreret som en del af Drupal

Systemet bliver ikke længere kaldt eller resultater heraf vist på dr.dk

Anbefalet handling

Tag API skal vi have kikket på i den generelle renovation af Relations systemer.

8.50 Ritzau

figure 49: `MATCH (x{name:'Ritzau'})--(y) RETURN x,y`

Kort beskrivelse

Eksternt system. Dataleverandør til Drupal-artikler. Tilbyder bred dækning af nyhedsbilledet i form af kortfattede telegrammer.

Anbefalet handling

Eksternt system som er kritisk for redaktionen. Ingen handling nødvendig.

8.51 Ratatosk

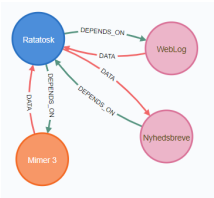

figure 50: `MATCH (x{name:'Ratatosk'})--(y) RETURN x,y`

Kort beskrivelse

RSS feed generator. Leverer data til Nyhedsbreve.

8.52 Drupal Vinr Mimer

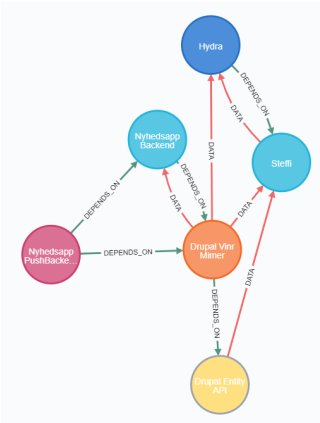

figure 51: `MATCH (x{name:'Drupal Vinr Mimer'})--(y) RETURN x,y`

Kort beskrivelse

Mikroservice til at udstille indhold ud fra flere forskellige identifikationsmuligheder

Anbefalet handling

I forbindelse med alignment med reference arkitekturen, så skal vi have kikket på om denne service har et eksistensgrundlag eller om den skal erstattes af en anden.

8.53 DRIP

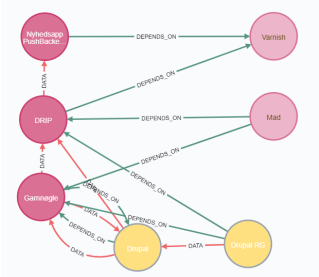

figure 52: `MATCH (x{name:'DRIP'})--(y) RETURN x,y`

Kort beskrivelse

DR Integrationsplatform, et køsystem til udveksling og transformation af beskeder mellem systemer. Har ind- og ud-adapters. Tilbyder applikationer at udveksle data pålideligt samt at distribuere til mange forskellige aftagere.

Anbefalet handling

Systemet ejes af Virksomhedssystemer, så vil ikke være oplagt at ændre på ud fra ønsker i Web & Apps. Skal muligvis erstattes af et Event system alt efter hvad analysen af det viser.

8.54 Global Navigation

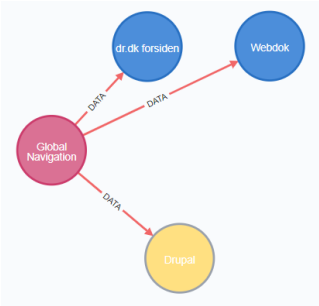

figure 53: `MATCH (x{name:'Global Navigation'})--(y) RETURN x,y`

Kort beskrivelse

Global top navigation og footer. Script der indsætter navigation og footer på dr.dk.

Anbefalet handling

Ingen umiddelbart nødvendig.

8.55 BrugerUpload

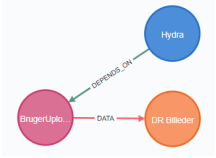

figure 54: `MATCH (x{name:'BrugerUpload'})--(y) RETURN x,y`

Kort beskrivelse

React komponent med egen backend, der bl.a. bruges til upload af brugernes egne vejrbilleder. Har sin egen Drupal komponent.

Anbefalet handling

Ingen umiddelbart nødvendig.

8.56 Imagescaler

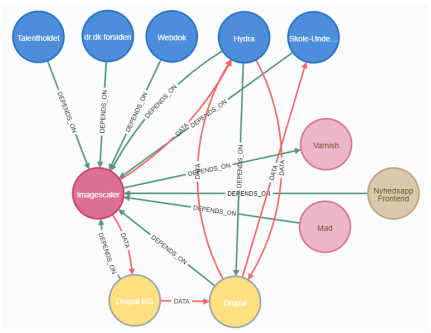

figure 55: `MATCH (x{name:'Imagescaler'})--(y) RETURN x,y`

Kort beskrivelse

Egenudviklet billedskaleringsservice, som Drupal RG og frontend anvender til at skalere billeder efter behov.

Billedskalering, konvertering og systemspecifikke workarounds for at aftage billeder.

Anbefalet handling

Der er ingen udestående problemer med Imagescaler som den står nu.

8.57 ADFS

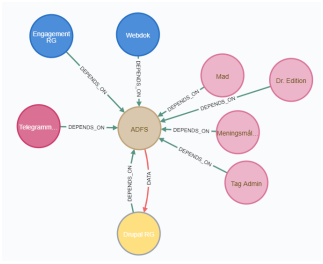

figure 56: `MATCH (x{name:'ADFS'})--(y) RETURN x,y`

Kort beskrivelse

Active Directory Federation Service, SSO for DR-brugere.

SSO-loginservice til DR's ansatte, så de kan anvende samme login til alle systemer.

Anbefalet handling

Der er ingen udestående problemer med ADFS som den står nu.

8.58 Global Assets

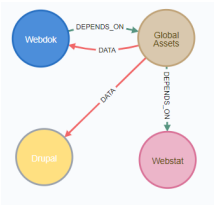

figure 57: `MATCH (x{name:'Global Assets'})--(y) RETURN x,y`

Kort beskrivelse

Globale JavaScript og CSS assets, primært til gammelt dr.dk design.

Anbefalet handling

Da Global Assets primært benyttes af det gamle design, så virker det oplagt at vi kikker på om det giver mening at bibeholde de gamle sider. Det kan meget vel give mening at få ryddet lidt op og her kunne være et oplagt sted at starte.

8.59 Global

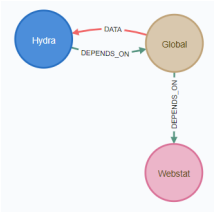

figure 58: `MATCH (x{name:'Global'})--(y) RETURN x,y`

Kort beskrivelse

Globale front-end assets (JS, CSS, fonte), mere strømlinet erstatning for Global Assets. Hostes på Heroku.

Anbefalet handling

Der er ingen udestående problemer med Global som den står nu.

8.60 Webstat

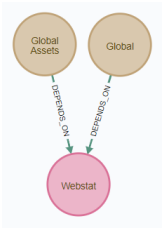

figure 59: `MATCH (x{name:'Webstat'})--(y) RETURN x,y`

Kort beskrivelse

Diverse statistik (mv.) scripts. Det meste af funktionaliteten håndteres i dag af Ensighten men f.eks. Gallup håndteres stadig af Webstat.

Anbefalet handling

I oprydningens ånd, så skal vi have kikket på om det er relevant at få Webstat pensioneret og de sidste anvendere flyttet til f.eks. Ensighten.

8.61 Describe Assets

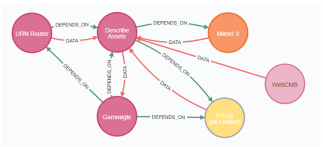

figure 60: `MATCH (x{name:'Describe Assets'})--(y) RETURN x,y`

Kort beskrivelse

Udstiller data for givne indholdsressourcer (tre separate services for hvert større indholdssystem).

Ud fra en URN kan Describe Assets udstille de relevante data for denne URN, som en hurtig og endpoint-agnostisk service der omsætter et ID til data

Anbefalet handling

Der er ingen udestående problemer med Describe Assets som den står nu. Når vi kikker nærmere på vores generelle Datamodel og URN / URL, så kan det meget vel være der også kommer opgaver til Describe Assets.

8.62 URN Router

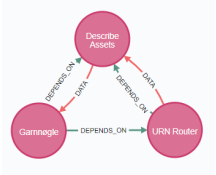

figure 61: `MATCH (x{name:'URN Router'})--(y) RETURN x,y`

Kort beskrivelse

URN-oversætter, der returnerer links til Describe Assets.

For at afskærme aftagere fra hvilke systemer der udstiller hvilke typer indhold, tilbyder URN Router ét samlet endpoint hvortil en URN kan sendes, og der returneres en URL til et endpoint hvor de ønskede data om den givne URN kan hentes.

Anbefalet handling

Der er ingen udeståender problemer med URN Router som den står nu. Når vi kikker nærmere på vores generelle Datamodel og URN / URL, så kan det meget vel være der også kommer opgaver til URN Router.

8.63 Freja

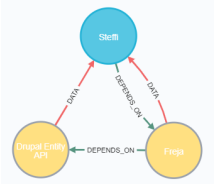

figure 62: `MATCH (x{name:'Freja'})--(y) RETURN x,y`

Kort beskrivelse

Drupal 8 site, præsentation og forside API.

RG samt API til vedligeholdelse og udstilling af præsentationsdata vedr. Drupal sektioner, forsider og artikelsider, f.eks. farver, logoer, backdrops.

Anbefalet handling

Der er ingen udeståender problemer med Freja som den står nu.

8.64 Karrierekanonen

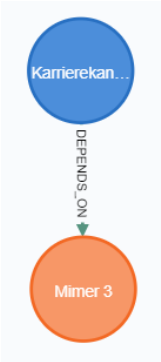

figure 63: `MATCH (x{name:'Karrierekanonen'})--(y) RETURN x,y`

Kort beskrivelse

Applikation til musik, præsentation (Front-end + admin)

Anbefalet handling

Der er ingen udeståender problemer med Karrierekanonen som den står nu.

8.65 Apigee

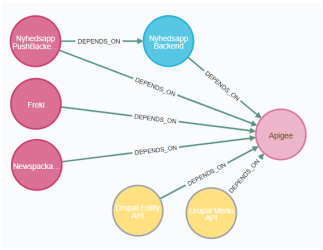

figure 64: `MATCH (x{name:'Apigee'})--(y) RETURN x,y`

Kort beskrivelse

API-gateway for DR's backendsystemer hostet udenfor DR Byen.

API-gateway, der afskærmer backendsystemer fra DDoS-angreb, tilbyder API-authentication og nøglehåndtering.

Anbefalet handling

Det giver rigtigt god mening at alle de APIer vi udstiller ligger bag en god API-gateway. Vi skal have kikket på præcist hvad vi gør med API gateway når vi begynder at nærme os at flytte systemer over på pipeline

8.66 Varnish

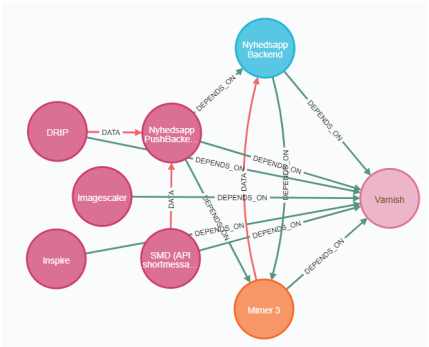

figure 65: `MATCH (x{name:'Varnish'})--(y) RETURN x,y`

Kort beskrivelse

Håndterer caching og routing, primært for systemer hostet i DR Byen.

Varnish sikrer at forespørgsler til backends eller andre services rammer et centralt sted, og derefter sendes til den relevante service eller backend, samt tilbyder cachingmuligheder for svaret, således at belastningen på service eller backend reduceres kraftigt.

Anbefalet handling

Varnish er i sig selv ikke noget vi kan påvirke i Web & Apps. Vores afhængighed af systemet hænger sammen med afhængigheden af de systemer der ligger bag Varnish. De systemer der benytter Varnish i dag er enten nogen der allerede er planer for at erstatte eller ligger i systemer som vi ikke kan påvirke.

8.67 Freki

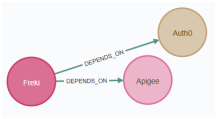

figure 66: `MATCH (x{name:'Freki'})--(y) RETURN x,y`

Kort beskrivelse

System til at gemme applikations specifik personfølsom data.

Anbefalet handling

Der er ingen udestående problemer med Freki som den står nu.

8.68 Berigelsessider

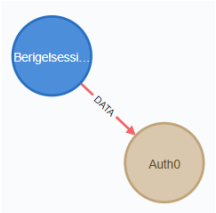

figure 67: `MATCH (x{name:'Berigelsessider'})--(y) RETURN x,y`

Kort beskrivelse

Opsamler ekstra data om en bruger i forbindelse med Login.

Anbefalet handling

Der er ingen udestående problemer med Berigelsessider som den står nu.

8.69 IronMQ

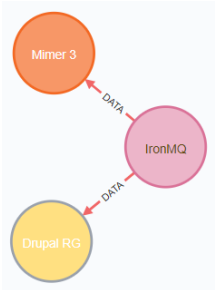

figure 68: `MATCH (x{name:'IronMQ'})--(y) RETURN x,y`

Kort beskrivelse

Køsystem til tilstandsløs integration mellem systemer, primært udenfor DR Byen.

Varetager modtagelse og distribution af beskeder mellem systemer, kan underrette aftagende systemer når der er nyt indhold i en given kø, fremfor at disse systemer poller IronMQ.

SaaS implementering, så dermed ikke noget vi kan udvikle på selv.

Anbefalet handling

En eventbaseret tilgang til opdatering af systemer er i mange tilfælde en strålende løsning og gerne også en vi skal have kikket nærmere på. Om vi skal benytte IronMQ eller et andet pub / sub system skal vi have kikket nærmere på.