SaaS - Software-as-a-service

Literatuuronderzoek

Jens van Lierop
Laméco Development BV
08-12-2021

Versiebeheer

Versie	Datum	Opmerking
0.1	14-10-2021	Eerste opzet
0.2	26-10-2021	SaaS uitgewerkt
1.0	29-10-2021	Project Digital + Conclusie uitgewerkt
1.1	15-11-2021	Toevoegen bronverwijzingen, Toelichting SaaS
1.2	08-12-2021	Toevoegen ureninschatting

Inhoudsopgave

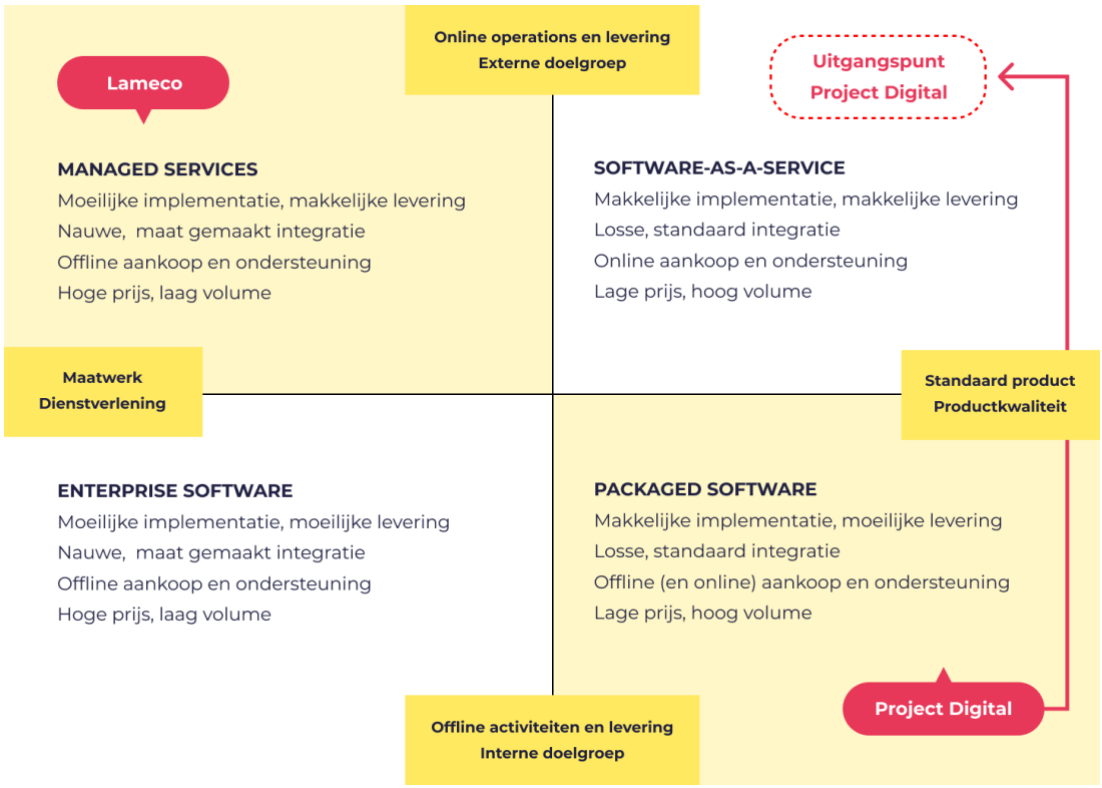
Inleiding

Dit onderzoeksverslag staat in het teken van Software-as-a-service

Wat is Saas?
Software-as-a-service (SaaS) is een softwaredistributiemodel, waarbij software centraal wordt gehost en op abonnementsbasis wordt aangeboden aan de gebruiker. SaaS staat ook bekend als "on-demand software" en webgebaseerde/webgehoste software.

SaaS is onderdeel van cloudcomputing, dit is een groep services waarbij computersysteembronnen on-demand beschikbaar worden gesteld, met name gegevensopslag en rekenkracht, zonder dat deze actief beheerd hoeven worden door de gebruiker zelf. Onder cloud computing vallen services zoals infrastructuur as a service (IaaS), platform as a service (PaaS), desktop as a service (DaaS). Deze services nemen in toenemende mate het beheer uit handen van de gebruiker en managen alles vanuit een centrale plek (de cloud).

Waarom SaaS?
Om zoveel mogelijk gebruikers aan boord te krijgen van Project Digital is de online beschikbaarheid van de software een topprioriteit. Beschouw het onderstaande figuur, Project Digital bevindt zich momenteel rechtsonder in de matrix. De software wordt enkel offline aan bestaande klanten aangeboden, waardoor de software een niet heel groot bereik heeft, omdat het moeilijk aan te bieden is.



SaaS: Alignment matrix – Notitie: Samenvoeging en adaptatie 'Technology, economics, business model en customer need alignment' diagrammen (York, 2017)

Als bedrijf zelf bevindt Laméco zich linksboven in onder ‘Managed Services’. Ze ontwikkelen maatwerk oplossingen voor hun klanten en richten voor iedere klant een omgeving in die door Laméco wordt onderhouden. Dit heeft als grote voordeel dat de omvang van de projecten vooraf bekend zijn en dat er relatief weinig ondersteuning en onderhoud nodig is, omdat het enkel door die klant gebruikt wordt. Als nadeel heeft het echter dat je maar eenmalig geld verdient.

Aan de andere kant van het spectrum bevindt zich Software-as-a-Service (SaaS). Deze platforms worden voor een langere tijd doorontwikkeld, onderhouden en online aangeboden als abonnementsvorm voor een bepaald bedrag. Hierdoor genereer je maandelijks vaste omzet en kun je de software gemakkelijk aan meer gebruikers aanbieden. De nadelen van SaaS zijn dat je als bedrijf tijd kwijt voor de ondersteuning van dat enkele product, omdat er veel meer gebruikers van het systeem zijn en dat je oplossing, vanwege het variabele aantal gebruikers, schaalbaar moet zijn.

Om een grotere doelgroep te bereiken en daarmee het volledige potentieel van Project Digital te bereiken is de transitie van packaged product naar SaaS noodzakelijk. Daarom zal er in dit document onderzoek worden gedaan naar deze vorm van softwaredsitributie.

Doelen

- **Kennis opdoen over SaaS**
  Algemene kennis vergaren over SaaS en ontdekken met welke zaken er rekening gehouden moet worden als er wordt gemigreerd naar een dergelijk model.

- **Vergelijking Project Digital**
  Achterhalen op welke aspecten Project Digital op dit moment tekortkomt op SaaS gebied en welke wijzigingen er nodig zouden zijn om het tot een ware SaaS applicatie te maken.

Onderzoeksvraag

Wat is ervoor nodig om Project Digital om te bouwen tot een SaaS-oplossing?

- Hoe ziet de huidige architectuur van Project Digital eruit? Welke invloed heeft de architectuur op de SaaS implementatie?
- Wat is er op technisch vlak, zowel in het kader van software als infrastructuur, nodig om een bestaande software om te bouwen tot een SaaS?
- Wat zijn de voor- en nadelen van verschillende implementaties?
- Welke functionaliteiten op het gebied van SaaS zijn al (gedeeltelijk) gebouwd of uitgewerkt? Welke missen er nog?

Methodiek

Literature study
In dit onderzoek wordt er via online zoekopdrachten gezocht naar relevante informatie die betrekking heeft op het migreren naar een SaaS platform.

“Make a ‘search plan’; identify relevant keywords. Find and judge your sources. Be sure to determine the relevance and reliability of a source by examining the publication date ,the author’s background and whether the author has a commercial interest.”
(HAN University of Applied Sciences, z.d.)

Aanpak

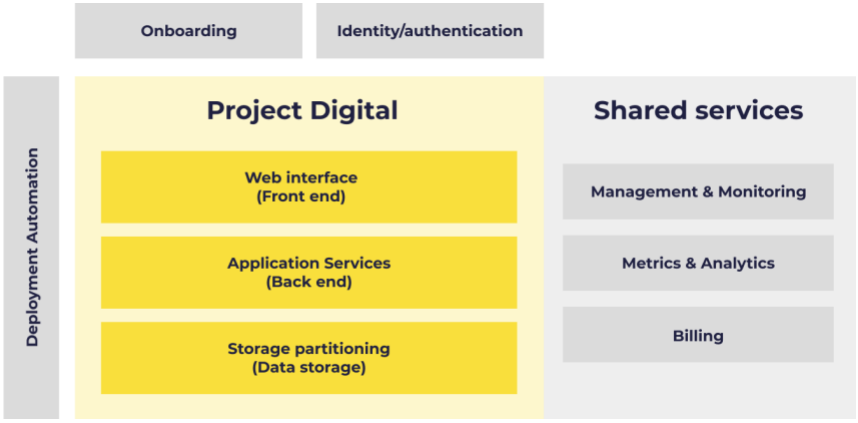
1. Er wordt online gezocht naar ‘SaaS migratiemodellen’ in zowel commerciële (Google) als academische zoekmachines (Biep.nu, Researchgate.net). De benodigde informatie dient globaal en praktisch toepasbaar te zijn; Informatie die gebonden is aan specifieke programmeertalen of infrastructuren is mogelijk niet van toepassing.
2. Er wordt specifiek gezocht naar videos, rapporten of publicaties die eerstehands informatie verstrekken; Typische content marketing publicaties worden uitgesloten. Enkel publicaties uit academische bronnen of informatie van toonaangevende instanties op het gebied van software ontwikkeling en cloud hosting worden gebruikt. Daarnaast kunnen secundaire bronnen (bronnen van bronnen) worden geraadpleegd, mits deze aan de voorgenoemde criteria voldoen.
3. Vervolgens worden de ontwikkelaars van Project Digitaal bevraagd over de huidige opbouw en functionaliteit van Project Digital en welke overwegingen een rol hebben gespeeld bij de implementatie. De online gevonden informatie dient als basis voor deze gesprekken.
4. De verschillen tussen de gewenste situatie en de huidige situatie worden in kaart gebracht en op een rij gezet.

Resultaten

SaaS architectuur

Om meer te begrijpen over SaaS architectuur, is de term 'tenant' erg belangrijk:
Een tenants (letterlijk: huurder) is een organisatie, bedrijf of persoon die gebruik maakt van je systeem, dit kan een enkele gebruiker zijn of een grotere groep van gebruikers.

In tegenstelling tot traditionele software distributiemodellen wordt er bij het Software-as-Service model niet voor elke tenant een eigen omgeving ingericht, maar maken meerdere tenants gebruik van hetzelfde systeem. Je spreekt dan over multitenancy. Multitenancy voegt een extra laag van complexiteit toe, omdat je niet langer te maken hebt met een systeem-gebruiker structuur maar met systeem-tenant-gebruiker.



Project Digital SaaS architectuur – Notitie: Adaptatie van 'SaaS architecture' diagram (Golding, 2020)

Om een bestaande applicatie uit te kunnen rollen als een SaaS-product, is er meer nodig dan alleen de oorspronkelijke software. De software moet namelijk omgeven worden door aanvullende services om voldoende voorzieningen te bieden voor de beheerders en tenants van het systeem.

Het bovenstaande figuur geeft in het gele blok de software zelf weer. De omringende grijze blokken geven de aanvullende services weer. Deze services bestaan uit:

- **Onboarding:** voor het in werking stellen van alle processen die nodig zijn voor het aanmelden van nieuwe tenants.
- **Automatische deployment:** voor het aanmaken en uitrollen van omgevingen voor nieuwe tenants.
- **Identity/Authentication:** voor het koppelen van gebruikers aan tenants en het authenticeren van gebruikers voor de juiste omgeving.
- **Management and monitoring:** voor het beheren van alle tenants binnen het systeem (administrator) en voor het opsporen van fouten.
- **Metrics & Analytics:** voor het meten van de status van de individuele tenants en de impact die ze hebben op het systeem als geheel.
- **Billing:** voor automatiseren van betalingen, waarbij eventueel het gebruik van resources (zie voorgaand punt) wordt gebruikt voor het maken van een prijsopgave.

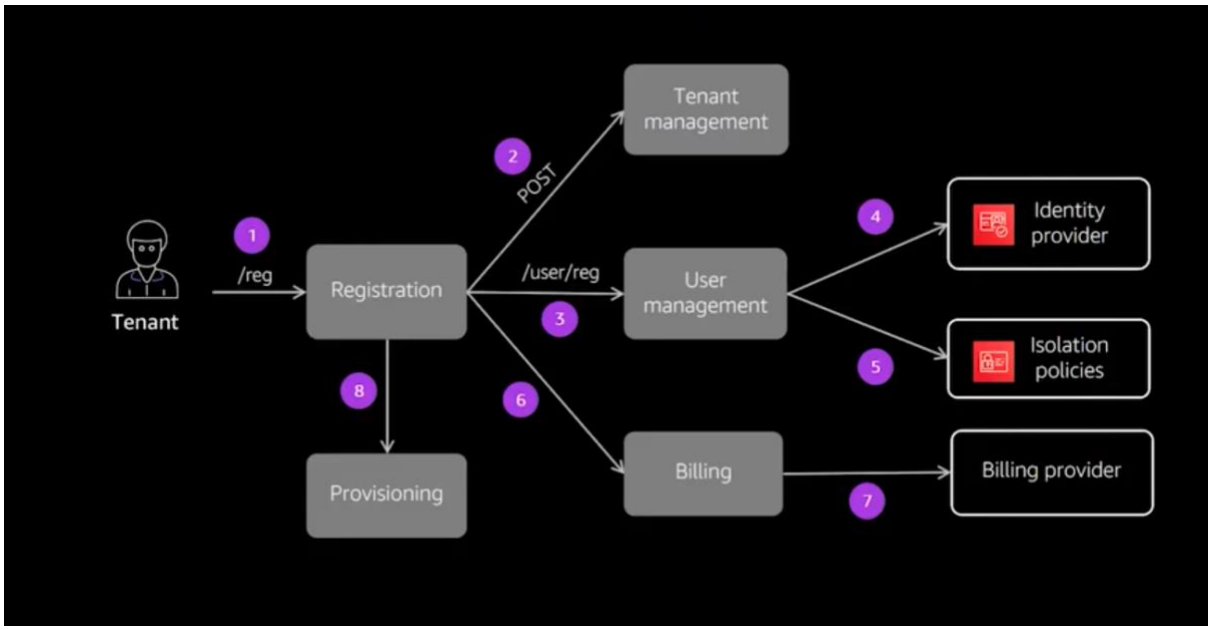
Behalve bovenstaande punten is tenant isolation, het scheiden van tenants binnen één systeem, een kernonderdeel van SaaS. Alle punten worden in de volgende hoofdstukken behandeld.

Onboarding

Een gestroomlijnd onboardingsproces is cruciaal voor het creëren van een instapvriendelijke, schaalbare SaaS omgeving. Het zet namelijk alle processen in werking die nodig zijn voor het aanmaken van nieuwe tenants en hun bijbehorende omgevingen.

Wanneer je elk van deze stappen geautomatiseerd hebt, kunnen nieuwe tenants vrijwel direct gebruik maken van je systeem en maakt het niet uit hoeveel nieuwe gebruikers je krijgt (afhankelijk van de schaalbaarheid van je onderliggende infrastructuur).

Een typish onboardingsproces ziet er als volgt uit:



SaaS onboarding - (Golding, 2021)

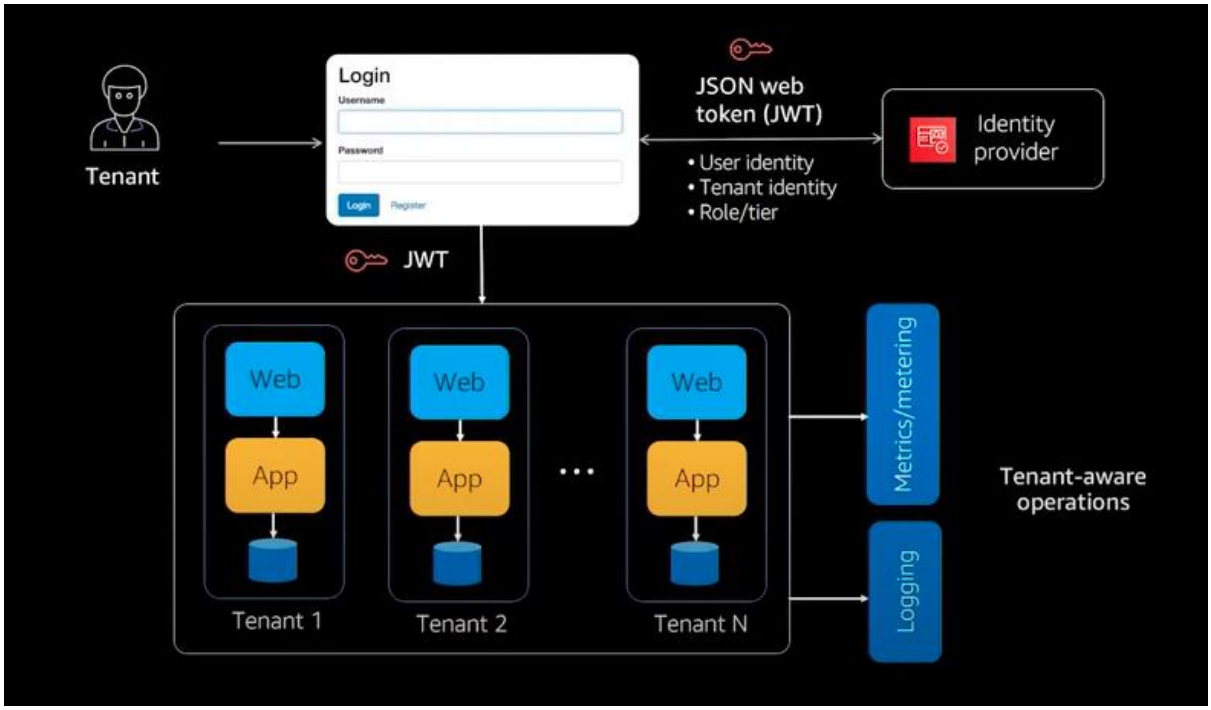
1. Het onboarding proces wordt in werking gezet wanneer een tenant naar een registratiepagina navigeert of zich via vergelijkbare manier aanmeldt. De basisgegevens worden door de nieuwe tenant opgegeven.
2. Een nieuwe tenant wordt aangemaakt in de tenant manager. Hierin wordt alle informatie over tenants in opgeslagen, zoals billing gegevens of configuraties. De zojuist aangemaakte tenant wordt teruggegeven aan de registratiemanager.
3. Voor deze tenant wordt een eerste nieuwe gebruiker toegevoegd aan het systeem. Deze eerste gebruiker wordt de administrator van desbetreffende tenant.
4. De nieuwe gebruiker wordt aangemaakt en vastgelegd in een identity provider. Behalve de de standaardgegevens wordt door middel van custom velden de verbinding tussen de gebruiker en tenant vastgelegd.
5. Vervolgens wordt het toegangsbeleid (Acces/isolation policies) voor de tenant geconfigureerd, zodat deze alleen toegang krijgt tot de eigen omgeving en data.
6. Als laatste wordt er een verbinding opgezet met een billing provider en wordt er een billing account aangemaakt voor de tenant.
7. Dit is meestal een extern systeem, zoals Stripe, maar het kan ook een intern systeem zijn.
8. Tegelijkertijd wordt er een omgeving opgetuigd voor de gebruiker. De benodigde infrastructuur wordt aangelegd en de juiste routing wordt geconfigureerd, zodat tenants naar hun omgeving kunnen navigeren.

Identity and Access Management (IAM)

IAM heeft alles te maken met authenticatie en autorisatie. Authenticatie is het valideren van de identiteit van een gebruiker, autorisatie is het verlenen van toegang aan een bepaalde gebruiker tot een specifieke groep resources, waar deze persoon toegang tot hoort te hebben.

IAM-systemen komen voor in vele verschillende vormen en de vereisten van een dergelijk systeem hangen af van de software waarin het toegepast wordt. Gebruikelijk is er al een vorm van IAM ingebouwd in bestaande applicatie, maar omdat er in SaaS-applicaties de tenant context als extra laag tussen het systeem en de gebruikers in ligt is het verhaal iets complexer.

Onderstaande figuur geeft een typische authenticatie en autorisatie flow weer.
Een gebruiker logt in met zijn/haar gegevens. Wanneer de authenticatie succesvol is wordt er door de identity provider een JSON Web Token (JWT) teruggeven.



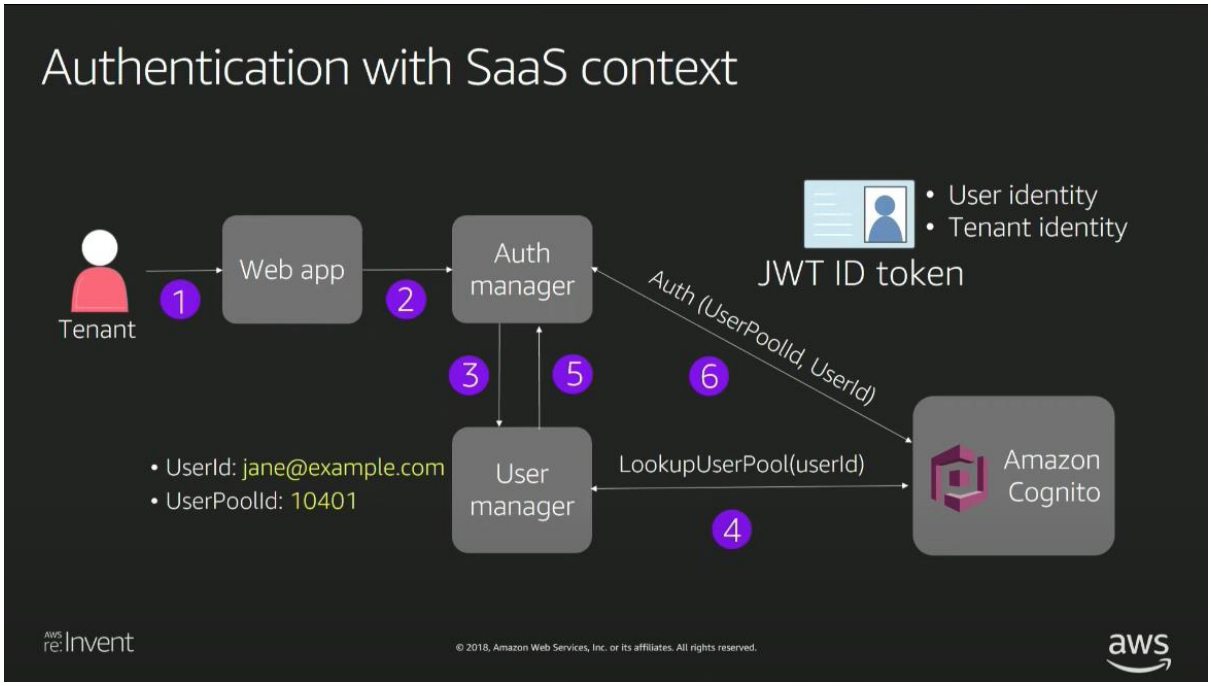
Authenticatie met tenant context - (Golding, 2021)

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. (Auth0® Inc., z.d.)

Dit autorisatietoken bevat zowel identiteit van de tenant als de identiteit en rol van de gebruiker. Het dient als het ware als een digitaal identeitsbewijs. Voor alle verschillende lagen van de applicatie kan deze identiteit afgewogen worden tegen de ingestelde runtime access policies (toegangsregels), om te bepalen of toegang tot mogelijk is.

De 'tenant context' in de JWT geeft niet alleen toegang tot de resources van de tenant, het zorgt ook voor de nodige context binnen de gedeelde services. Zaken die specifiek zijn aan iedere tenant (zoals logs, verbruik, metrics en analytics) kunnen vastgelegd worden met behulp van de info uit de JWT. (Golding, 2021)

Voorafgaand hieraan is het authentiseren van de gebruiker. Zoals eerder aangegeven verschilt het authenticatieproces van reguliere SaaS applicaties vanwege de tenant context. De onderstaande afbeelding geeft een voorbeeld van hoe dit proces kan verlopen.



Authenticatie met SaaS context - (Golding, 2018)

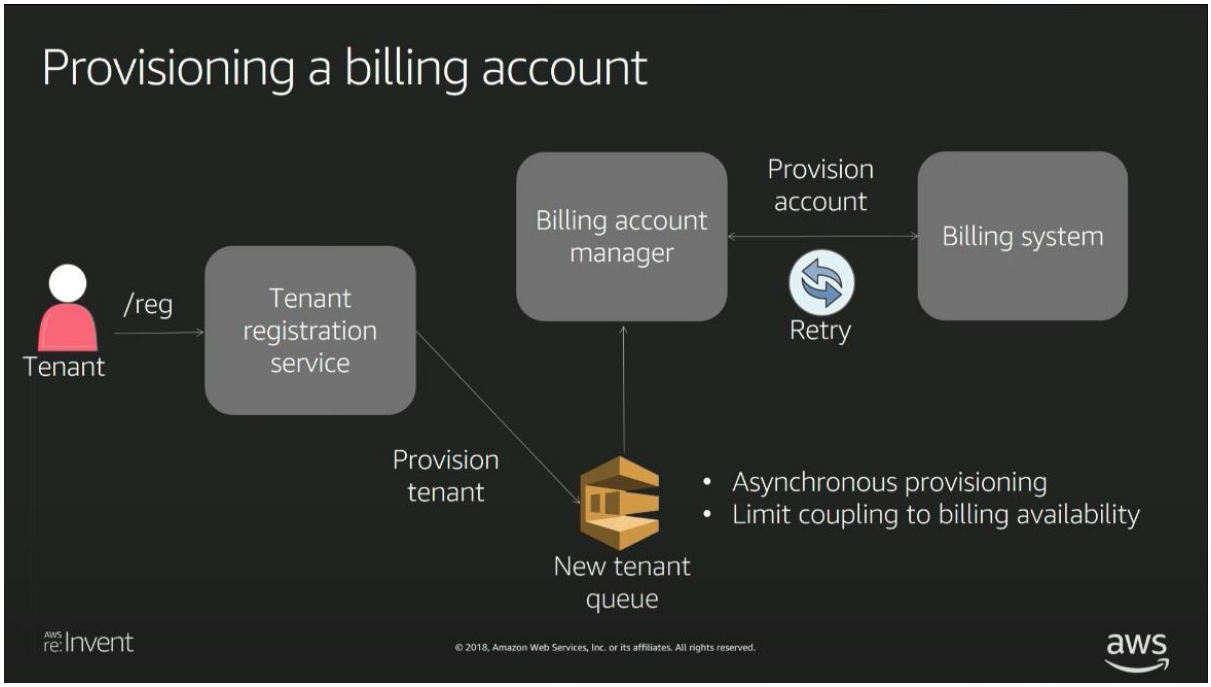
1. De gebruiker voert zijn/haar inloggegevens in de webapplicatie
2. Omdat het niet bekend is tot welke tenant de gebruiker behoort, kan er geen directe authenticatie plaatsvinden. Eerst moet bepaald worden tegen welke gebruikerspool de gebruiker geauthentiseerd moet worden. Om dit proces te orkestreren wordt een authenticatie manager gebruikt
3. De authenticatie manager geeft het id van de gebruiker door aan de user manager
4. De user manager haalt de juiste gebruikerspool van de matchende tenant op uit de identity provider.
5. De gebruikerspool wordt geretourneerd aan de authenticatie manager.
6. De gebruiker wordt in de identity provider tegen de juiste gebruikerspool geauthenticeerd. Er wordt een JWT gegeneerd en teruggegeven aan de gebruiker.

Billing

In tegenstelling tot een product uit de schappen kunnen SaaS-applicaties innovatieve verdienmodellen hebben. De prijs kan worden bepaald aan de hand van:

- Het aantal gebruikers
- De hoeveelheid gebruik
- Vast bedrag per maand/jaar
- Freemium model (betaalde aanvullingen)

Om onnodig handmatig boekwerk te voorkomen zal er in alle gevallen een billing provider nodig zijn die automatisch een rekening aanmaakt voor de gebruiker. Onderstaand diagram geeft een manier weer waarop dit geïmplementeerd zou kunnen worden.



Voorzien van een billing account - (Golding, 2018)

Allereerst meldt een gebruiker zich aan, zoals eerder uiteengezet in stap 1 van het onboarding proces. Wanneer de tenant en gebruiker zijn aangemaakt wordt een tenant voorzien van een omgeving en tegelijkertijd wordt het billing acocunt aangemaakt. Meestal een extern billing systeem. Dit is een asynchroon proces, om te voorkomen dat het aanmaken van een tenant afhankelijk is van de beschikbaarheid van het billing systeem.

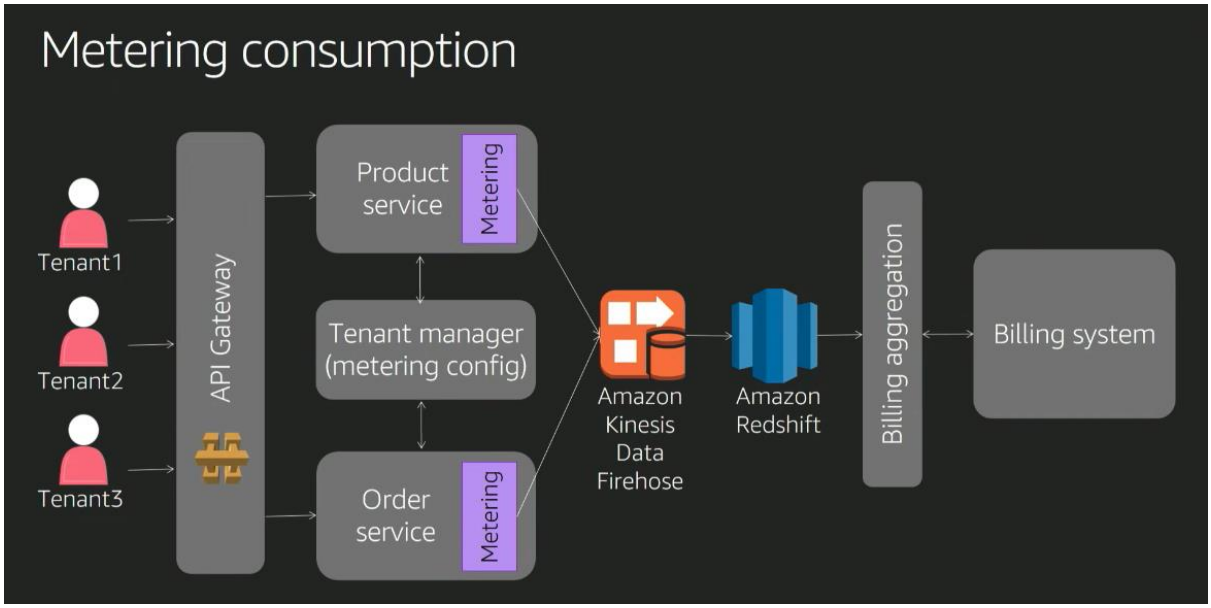
De tenant wordt toegevoegd aan een rij van tenants waarvoor het billing systeem moet worden gekoppeld. Deze billing systemen zijn meestal externe providers. Vervolgens wordt er voor deze tenant een billing account aangemaakt bij de provider, mocht dit niet lukken, dan is er een actie om op terug te vallen, in dit geval is het simpelweg opnieuw proberen.

In de billing accountmanager kan gebruiksdata van tenants worden vastgelegd en verstuurd naar de provider zodat er een rekening kan worden gemaakt. Zo kan er zelfs gewerkt worden met alternatieve verdienmodellen op basis van gebruik. Vanuit de billing account manager wordt de incassostatus van de tenant teruggegeven naar de tenant management service. Om deze bijvoorbeeld op inactief te zetten als een testperiode voorbij is, als er niet aan een betaling is voldaan.

Metrics & Analytics

Metrics & analytics heeft betrekking op het meten en inzichtelijk maken van de performance van het systeem en vooral elk van de individuele tenants. Ondanks dat het vaak als een bijzaak wordt gezien, is het net als in elk ander bedrijf essentieel voor de technische en operationele zaken en bepalend voor de bedrijfsvoering.

Werken met Metrics & Analytics is in principe een heel standaard proces en verschilt niet veel van gebruikelijke Business Intelligence (BI) Pipelines: Het bestaat namelijk uit logging (vastleggen van data), het verwerken van deze data en het analyseren en weergeven van de data in een BI tool. Uiteindelijk is het idee dat alle data overzichtelijk beschikbaar is in een dashboard waarin de verschillende stakeholders in één opzicht de status van het systeem en alle tenants kunnen zien. Voor het eerste onderdeel (logging) ben je vooral afhankelijk van je eigen implementatie, voor de overige twee onderdelen (verwerken en analyseren) kun je in de meeste gevallen, net als billing, het beste oplossingen van 3de partijen gebruiken.



Meten van verbruik - (Golding, 2018)

Inzicht in de individuele tenants is om de volgende redenen cruciaal:

Architectuur: Wijzigingen en optimalisaties in de architectuur om het maximale eruit te halen.
Operations: Er kan worden ontdekt hoe het systeem presteert onder de verschillende gebruikers. Vervolgens kan hier op worden ingespeeld door pro-actief misbruik van het systeem te voorkomen aan de hand van zichtbare patronen in het gebruik.
Kosten optimalisatie: Een van de primaire kostenposten van een SaaS driven business is IT-infrastructuur. Metrics kunnen worden gebruikt voor optimalisaties en besparingen op de infrastructuur.
Billing: Een onderdeel van Metrics & Analytics is metering, waarin het gebruik van het systeem per tenant wordt vastgelegd, zodat er een rekening gegenereerd kan worden.
Tiering strategy: De data kan ook gebruikt worden om te bepalen hoeveel je gaat vragen voor de verschillende features in je systeem en hoe je je tenants geleidelijk naar hogere tiers toe geleid.
Business modeling: Metrics kunnen zelfs invloed hebben op je businessmodel als geheel. Er kunnen mogelijkheden ontstaan die het bedrijf veel meer geld opleveren, of juist heel veel geld kosten. Deze moet je tijdig kunnen herkennen.

Afhankelijk van het type SaaS oplossing verschilt de implementatie van Metrics & Analytics, en daarmee de inhoud van het dashboard) enorm. Er is dan ook geen eenduidige, kant-en-klare oplossing die geïmplementeerd kan worden. Daarnaast heeft het weinig nut om alles tot in de kleine details te gaan meten, maar gaat het vaak om een grove schatting. Het is belangrijk om dat de complexiteit van de randservice altijd in verhouding staat met de waarde van de statistieken. (Golding, 2018)

De implementatie van Metrics en Analytics kan bereikt worden via een Business Intelligence workflow:

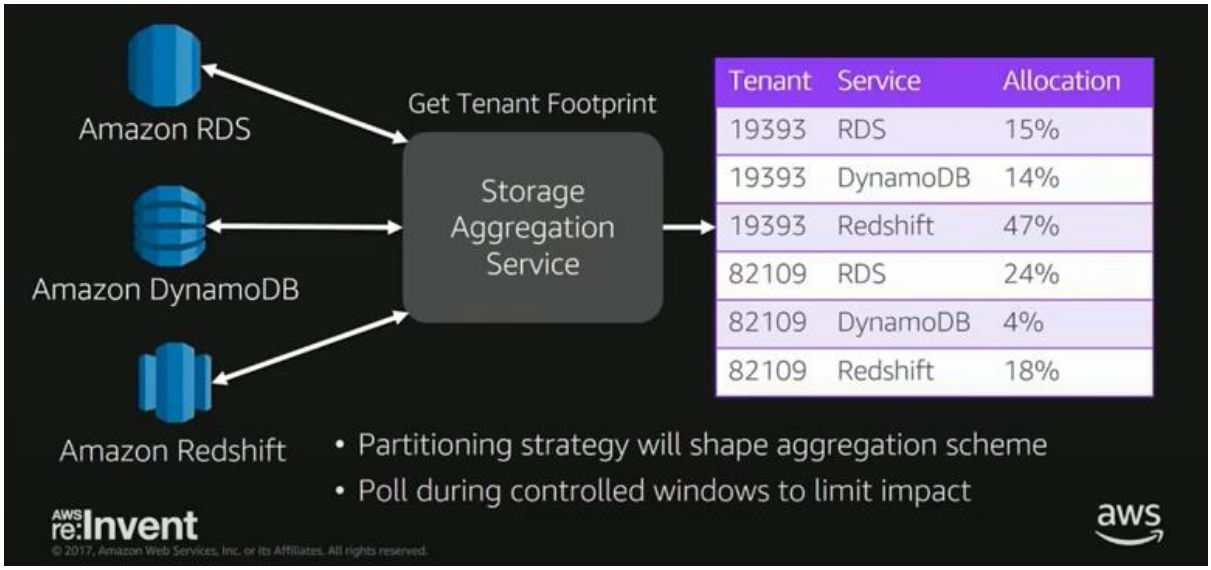
1. Allereerst moet er bepaald worden, welke Key Performance Indicators (KPI) belangrijk zijn voor de SaaS applicatie en het bedrijf eromheen.
2. Vervolgens moet er worden onderzocht welke data er nodig is om deze KPI's te kunnen weergeven en op welke manier de data wordt gevisualiseerd.
3. Daarna worden de juiste constructies gebouwd om deze data te verzamelen, verwerken en weer te geven.

Voor elke laag van de applicatie kunnen verschillende strategieën worden toegepast om de juiste data te verkrijgen. In de applicatie laag kan er bijvoorbeeld worden gekeken naar de volgende data:

- **Hoeveelheid aanvragen:** Hoeveel aanvragen worden er door de tenants gedaan, Naar welke routes worden deze aanvragen verstuurd.
- **Uitvoertijd:** Hoe lang is de server bezig met het uitvoeren en retourneren van tenant aanvragen.
- **CPU en Memory(RAM) impact:** Hoeveel procent van de resources wordt door elke tenant verbruikt (vergelijkbaar met taakbeheer in Windows)

Databases daarentegen zijn iets complexer en geven de volgende data:

- **Hoeveelheid lees- en schrijfbewerkingen:** Hoevaak wordt er data opgevraagd en weggeschreven naar de databse door iedere tenant.
- **Gebruikte oplsag:** Hoeveel opslag er in gebruik wordt genomen door iedere tenant. (Dit is een makkelijke metric, omdat de het gemakkelijk op te vragen is en maar een keer in de zoveel tijd hoeft te gebeuren).



Aggregatie van opslag data - (Golding, 2017)

De grootste afweging die gemaakt dient te worden is de keuze tussen het meten van frequentie en grootte. Zo kunnen 4 aanvragen even intensief zijn als 1000 aanvragen, wanneer de 4 aanvragen ieder 250x zo groot zijn als de andere 1000.

Tenant isolation strategy

Een van de belangrijkste aspecten op het gebied van SaaS is tenant isolation. Dit is de splitsing van gebruikers binnen 1 systeem, zodat ze niet bij elkaars gegevens kunnen. Het is vergelijkbaar met een appartementencomplex. Je wil dat alle bewoners toegang hebben tot het gebouw, hun eigen appartement en gedeelde voorzieningen (gas, water, licht), maar je wilt voorkomen dat ze elkaar appartement kunnen betreden.

Om tenants van elkaar te isoleren zijn verschillende strategieën mogelijk:

Silo model	**Pool model**
Een aparte omgeving voor elke tenant. **Voordelen** • Grove scheiding tussen tenants • Geen noisy neighbour problemen • Weinig aanpassingen aan bestaande code • Goede beheerbaarheid over individuele tenant omgevingen **Nadelen** • Minder flexibiliteit • Grotere kosten van infrastructuur • Slechte beheerbaarheid van het system als geheel	*Een gedeelde omgeving voor alle tenants.* **Voordelen** • Grotere flexibiliteit • Lagere kosten van infrastructuur • Hogere beheerbaarheid van het system als geheel **Nadelen** • Fijne scheiding tussen tenants • Veel aanpassingen aan bestaande code • Grotere kans op noisy neighbour problemen • Slechte beheerbaarheid van individuele tenant omgevingen
Bridge model	**Hybrid model**
Een omgeving waarin tenants een enkele onderdelen delen en andere onderdelen ieder voor zichzelf hebben. **Voordelen** • Incrementele veranderingen • Maar relatief weinig wijzigingen nodig aan de code **Nadelen** • Minder flexibiliteit • Moeilijk te managen	*Een combinatie van meerdere modellen waarin bepaalde tenants een hele eigen omgeving hebben terwijl andere tenants er een delen.* **Voordelen** • Tenants die meer betalen kunnen ondergeplaatst worden in een aparte omgeving • Makkelijker om te voldoen aan bepaalde SLA's of policies **Nadelen** • Moeilijk te managen door de combinatie van meerdere systemen

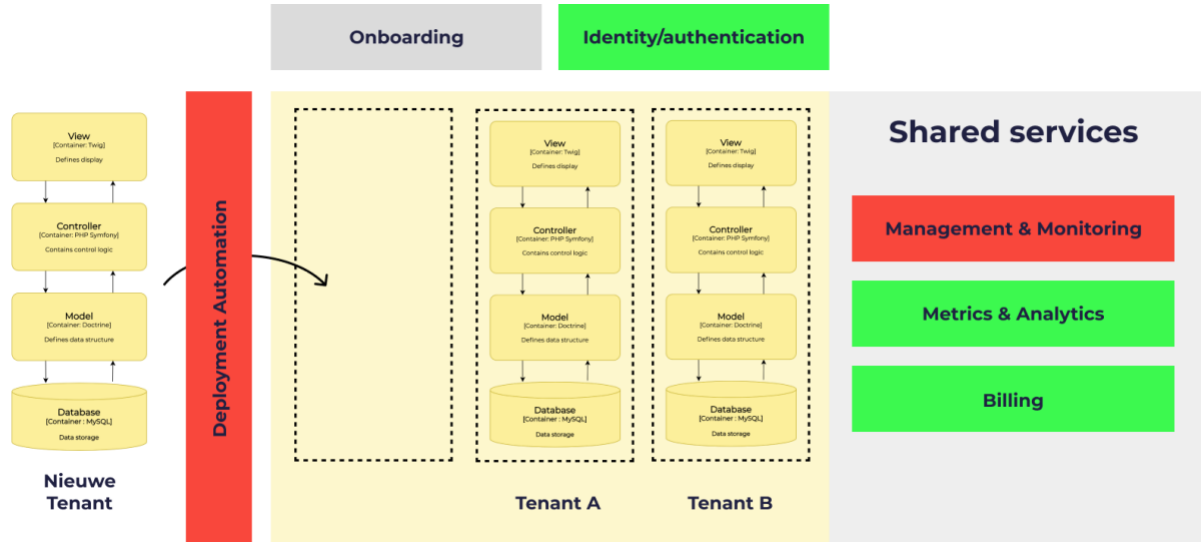
Tenant isolation strategies - (Amazon Web Services, Inc. & Golding, 2020), (Chong et al., 2006), (Krebs et al., 2015)

De keuze voor een bepaalde strategie is afhankelijk van een aantal factoren:

- **Tiering strategy:** Omgevingen kunnen worden ingericht aan de hand van wat klanten bereid zijn ervoor te betalen. Op basisniveau bestaat er dan volledig gedeelde environment, terwijl op professioneel niveau misschien volledig geïsoleerde environment wordt opgezet.
- **Noisy neighbour:** Bij gedeelde omgevingen kan de performance van tenants worden beïnvloed door andere tenants omdat alle voorzieningen gedeeld worden. Om dit te voorkomen kun je óf de gedeelde omgeving opschalen of volledige isolatie aanbieden door ieder zijn/haar eigen omgeving te geven.
- **Compliance requirements:** De strategie kan worden beïnvloed door de regulaties van de sectoren waarin je opereert. Om te voldoen aan de regelgevingen omtrent dataopslag en verwerking zijn sommige strategieën meer geschikt dan andere.
- **Legacy/orginele architectuur:** Afhankelijk van je bestaande software en hoeveel moeite er is om het is om deze om te bouwen.
- **Opportunities:** Specifieke business opportunities of vraagstukken vanuit (potentiele klanten) kunnen bepalend zijn.

Silo/isolated model

Bij een silo model wordt er gesproken over een indeling waarbij elke tenant een compleet eigen omgeving heeft. Er is dan ook sprake van grove isolatie, omdat de omgevingen via harde (infrastrucuur) grenzen van elkaar afgebakend zijn.



Project Digital silo model – Notitie: Adaptatie van 'Silo model' diagram (Golding, 2020)

Het systeem is gedecentraliseerd en opgesplitst in aparte accounts of vpc's/vm's (virtuele computers). Er wordt een aparte instantie van de volledige applicatie voor iedere tenant aangemaakt. Dit is doorgaans veel veiliger dan gedeelde omgevingen, omdat tenant niet door code exploits bij de data van andere tenants kunnen komen wanneer ze beperkt zijn tot hun eigen applicatie.

Ook zijn er minder wijzigingen nodig aan reguliere softwarestructuren, omdat je de bestaande stack opnieuw kan uitrollen voor een bepaalde tenant. Dit heeft echter wel als nadeel dat je voor elke nieuwe tenant een omgeving moet aanmaken en een nieuwe deployment moet doen. Management en monitoring wordt hierdoor ook lastiger, omdat je systeem als geheel is verspreid over allerlei verschillende omgeving.

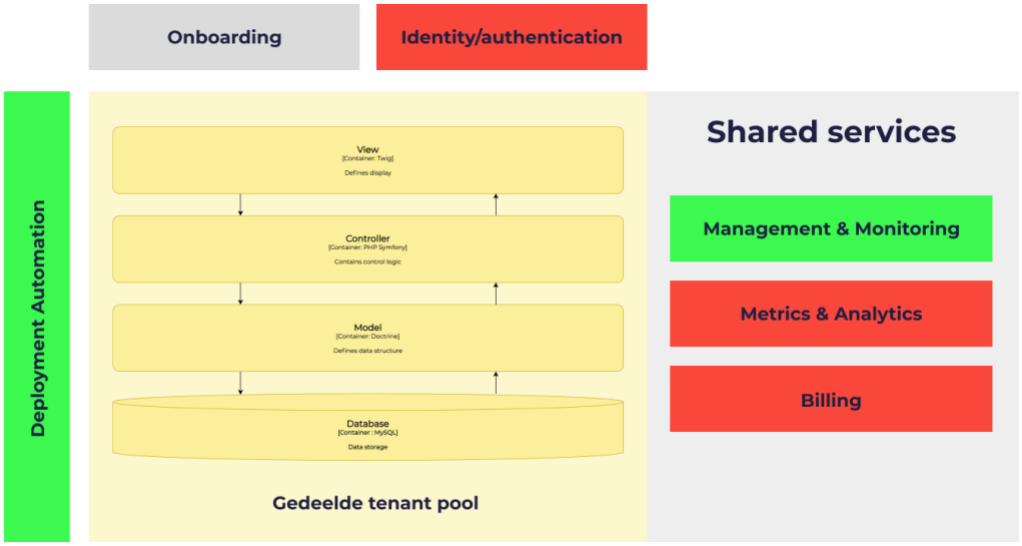
Wel is het makkelijker om inzicht te krijgen in elke individuele tenant omdat ze al opgesplitst zijn in aparte omgevingen. Zaken zoals metrics & analytics hoeven dan niet op applicatieniveau gescheiden te worden, maar kunnen op containerniveau worden ingezien. Ook IAM wordt makkelijker, omdat een gebruiker na het inloggen direct in zijn/haar eigen omgeving zit en daarom identificatie niet opnieuw bij elke stap nodig is. Enkel de rol van een gebruiker doet er dan nog toe.

Ondanks dat elke tenant een eigen stack heeft, zijn de omgevende functies voor elke tenant hetzelfde. Ook heeft elke tenant stack heeft dezelfde versie Zonder deze laatste twee punten is het geen ware SaaS maar meer een managed services model.

Dit model werkt vooral het best wanneer je weinig gebruikers hebt, of gebruikers hebt die een zware behoefte aan isolatie hebben, vanwege performance- of veiligheidsoverwegingen.

Pool/shared model

Bij een pool model worden alle voorzieningen gedeeld met alle tenants in één grote omgeving. In dit geval is er sprake van fijne isolatie, omdat er geen harde grenzen meer tussen de tenants zitten, maar in plaats daarvan op applicatieniveau de scheiding wordt gehandhaaft.



Project Digital pool model – Notitie: Adaptatie van 'Pool model' diagram (Golding, 2020)

In een omgeving waarin alles gedeeld wordt is IAM complexer dan bij een silo model. Voor elke aanvraag of bewerking van een gebruiker moet er gecontroleerd worden wie de gebruiker is, tot welke tenant deze persoon behoort en welke rollen hij/zij toegewezen heeft gekregen (Runtime scoped acces). Dit wordt gedaan door middel van de eerder genoemde runtime policies. Deze regels bepalen onder welke voorwaarden een gebruiker toegang krijgt tot een bepaalde resource. Vervolgens wordt er naar de JWT token van de gebruiker gekeken om te kijken of deze aan de voorwaarden voldoet.

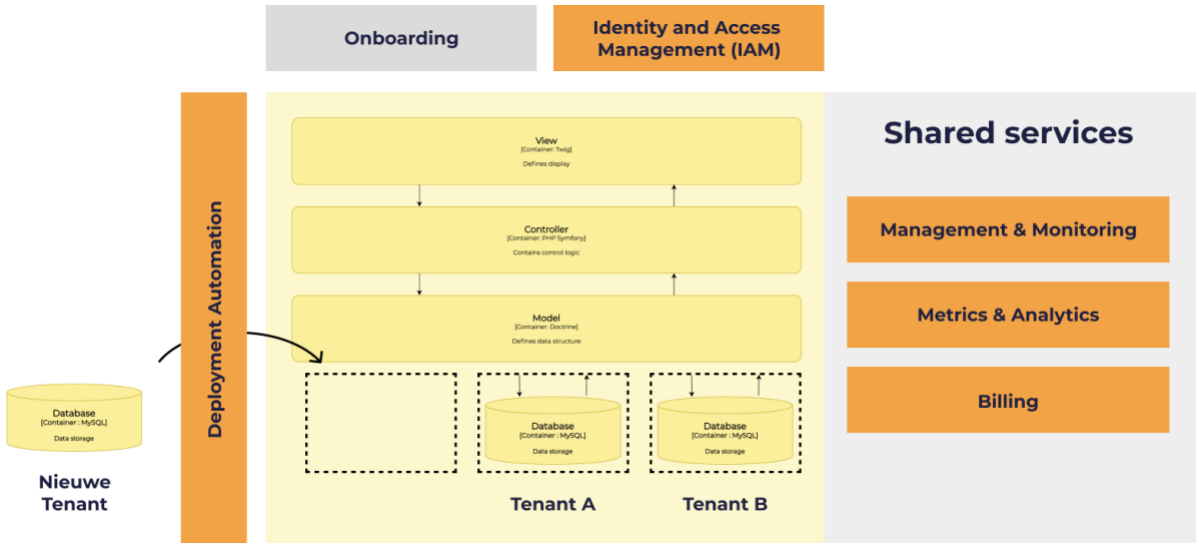
Niet alleen maken tenants gebruik dezelfde applicatie, ze delen ook de opslag, dus er zijn constructies nodig om de data goed te scheiden (data partitioning). Dit wordt in latere hoofdstukken verder behandeld. Een ander nadeel van is het monitoren en beheren van individuele tenants. Omdat alle tenants in één pool zitten, heb je constructies nodig om data van elke tenant eruit te filteren. Wederom komt hier de JWT token van pas (zie Identity and Access Management).

Ondanks dat de isolatie een stuk lastiger is, heeft het pool model wel een aantal grote voordelen: Allereerst is de deployment weinig tot geen werk. In plaats van dat er een hele nieuwe omgeving moet worden aangemaakt, wordt de tenant toegevoegd aan een bestaand systeem. Alleen de routing moet in orde worden gemaakt en de nieuwe tenant is gebruiksklaar. Daarnaast is het systeem vele malen flexibeler, omdat je maar met 1 omgeving werkt. Hierdoor is het monitoren, beheren en aanpassen van je gehele SaaS oplossing veel voor de hand liggender.

Dit model werkt vooral het best wanneer je veel gebruikers hebt, of gebruikers hebt die minder strikte eisen aan isolatie hebben.

Bridge/layered model

Het bridge model ligt tussen het silo en pool model in. Dit betekent dat sommige voorzieningen gedeeld worden met alle tenants en sommige delen compleet gescheiden zijn. Het bridge model kan op verschillende manieren worden toegepast omdat alle lagen van de applicatie (web, applicatie, data) kunnen worden opgedeeld of samengevoegd.



Project Digital bridge model – Notitie: Adaptatie van 'Bridge model' diagram (Golding, 2020)

Omdat het bridge model een combinatie van de voorgenoemde modellen is, liggen ook alle eerdergenoemde aspecten in het midden qua complexiteit.

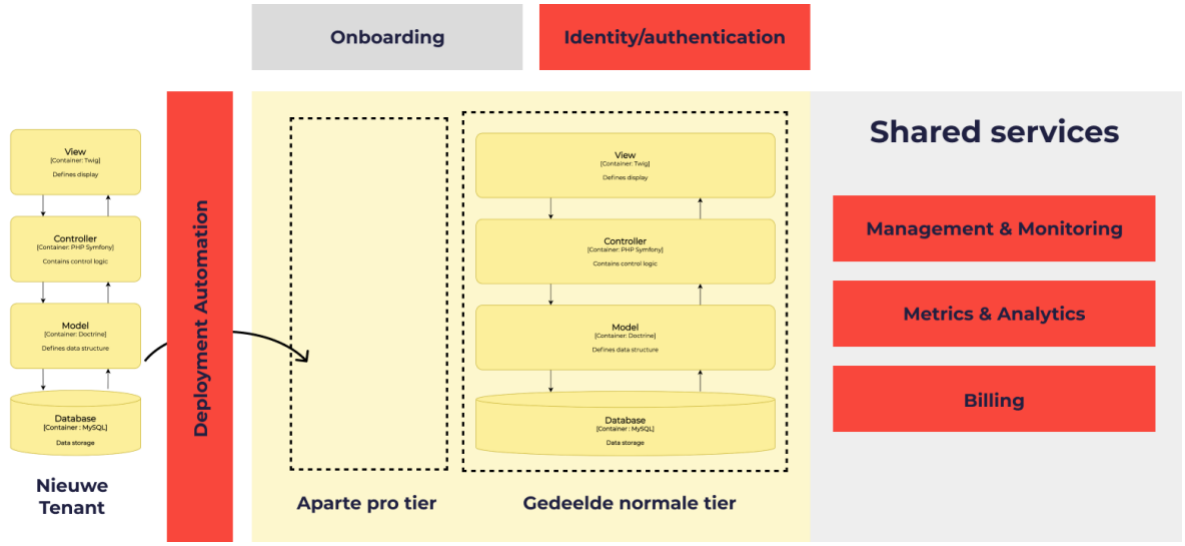
Het uitrollen van een nieuwe tenant omgevingen is makkelijker omdat er niet een gehele stack opgezet hoeft te worden, maar in plaats daarvan slechts een gedeelte wordt uitgerold. Er is dan wel nog steeds voor elke klant een database (in het bovenstaande geval).

Sommige gegevens kunnen hierdoor beter afgeschermd worden, waardoor er minder beveiligingsregels nodig zijn om ongeautoriseerde toegang te voorkomen. Ook kunnen bepaalde delen van de applicatie beter beheerd worden, terwijl dit bij andere zaken weer minder goed gaat. Zo zou bijvoorbeeld in het bovenstaande voorbeeld het aantal lees- en schrijfbewerkingen per database gemonitord kunnen worden, maar is het geheugenverbruik van de applicatie per tenant niet direct zichtbaar.

Met dit model kun je zonder enorme wijzigingen aan de code incrementele stappen zetten van een silo model naar een pool model (of andersom), maar daarmee offer je wel nog steeds veel flexibiliteit op. Het werkt het beste voor kleine- tot middelgrote gebruikersgroepen, die bepaalde behoefte hebben op het gebied van veiligheid of performance.

Hybride model

Als laatste model heb je een hybride model. Dit is een mix van meerdere modellen, waarbij sommige tenants een omgeving delen en andere tenants een compleet eigen omgeving hebben.



Dit model is vooral van toepassing wanneer je met verschillende gebruikersgroepen te maken hebt en het hangt sterk samen de 'tiering' van je applicatie. Gebruikers met hoge eisen aan veiligheid en performance kunnen voor een hogere prijs in een aparte silo ondergeplaatst worden, terwijl de rest van de gebruikers in een gedeelde omgeving terecht komt.

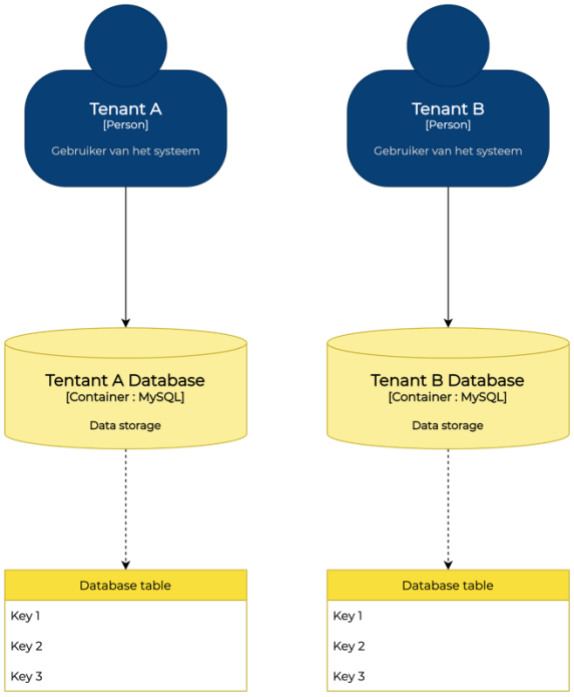
Op deze manier kan er worden voldaan aan de verwachtingen van de gebruikers en aan de SLA's (Service Level Agreements: De overeenkomsten die zijn gesloten met betrekking tot de kwaliteit van de werking van de software).

Alle aspecten van dit model zijn complex, omdat je met meerdere verschillende modellen tegelijk werkt. Er moet dus in alles worden voorzien en voor elke stap moet er gecontroleerd worden of het om een aparte of gedeelde omgeving gaat. Met name de beheerbaarheid van het systeem gaat achteruit, omdat elke stack anders is.

Dit model is voornamelijk bedoeld om specifieke, hogere klantsegmenten te kunnen bedienen.

Data partitioning strategy

In overeenstemming met de gekozen tenant isolation strategie, wordt er een passende data partitioning strategie toegepast.



Aparte database

De data kan voor iedere tenant in een aparte database worden opgeslagen. Dit is doorgaans de veiligste manier om de data van tenants op te slaan, aangezien je de individuele databases kan beveiligen tegen toegang door andere tenants. Ook zijn losse databse makkelijker aan te passen naar de behoefte van een specifieke klant.

Deze aanpak is wel het duurste en moelijkste als het gaat om onderhoud en wordt daarom ook vaak afgeraden.

Tenant isolation strategie
- Silo
- Layered

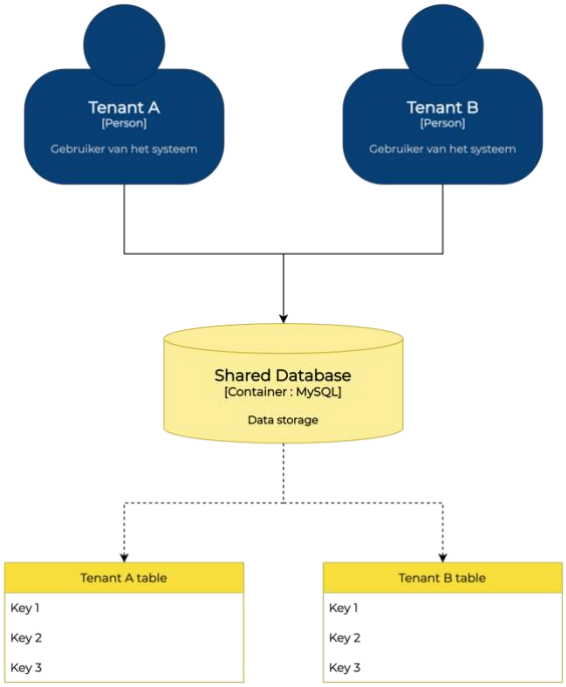
Gedeelde database, apart schema

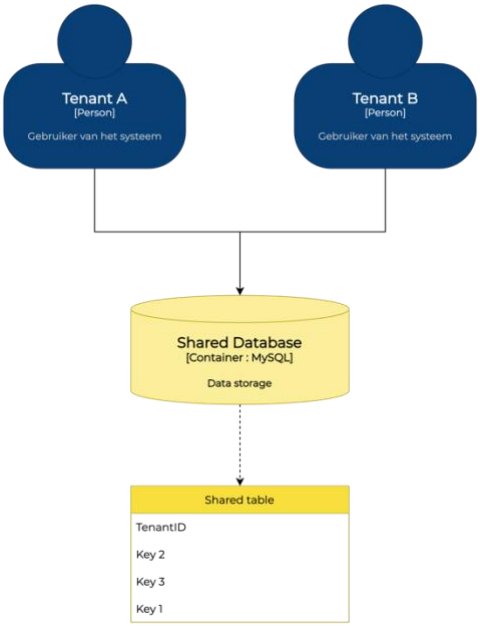
Bij deze aanpak wordt de data van de tenants in dezelfde database opgeslagen, maar heeft elke tenant wel een aparte set van tabellen.

Deze manier is relatief makkelijk te implementeren door specifieke user accounts toegang te verschaffen tot het juiste schema. Deze gebruikers kunnen vanuit dit account de tabelnaam aanroepen.

Ondanks dat het onderhouden van 1 database veel goedkoper en makkelijker is, is het nadeel van deze aanpak dat bij het herstellen van een database, de data van alle tenants moet worden teruggezet, waardoor er mogelijk data verloren gaat.

Tenant isolation strategie
- Pool
- Layered





Gedeeld schema, aparte keys

De laatste benadering omvat het gebruik van dezelfde database en dezelfde set tabellen om de gegevens van meerdere tenants op te slaan. De tabellen bevatten de records van verschillende tenants in willekeurige volgorde. Een tenant-ID kolom koppelt een record aan de juiste tenant.

Dit is de meest risicovolle manier om data op te slaan en het herstellen van individuele records vaak lastig. Wel is deze manier het makkelijkst te implementeren en het goedkoopst om aan te bieden en onderhouden. Je kan met deze aanpak namelijk per database de meeste tenants bedienen.

Tenant isolation strategie

- Pool
- Layered

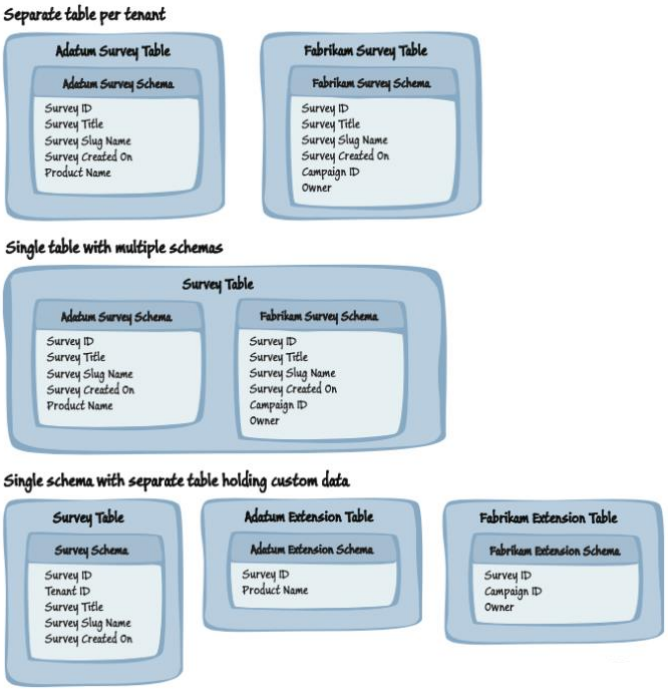
Data extensies

Ook wanneer tenants gebruik maken van verschillende datastructuren kunnen eerdergenoemde data partitioning strategieën gehandhaafd worden.

Wanneer je met aparte tabellen werkt, hebben ze per tenant allemaal een eigen datastructuur.

In gedeelde tabellen heb je de keuze een tabel met meerdere schema's te maken (als het ware meerdere varianten van dezelfde tabel), of om data extensies op te nemen in een aparte tabel die terugverwijst naar de hoofdtabel.

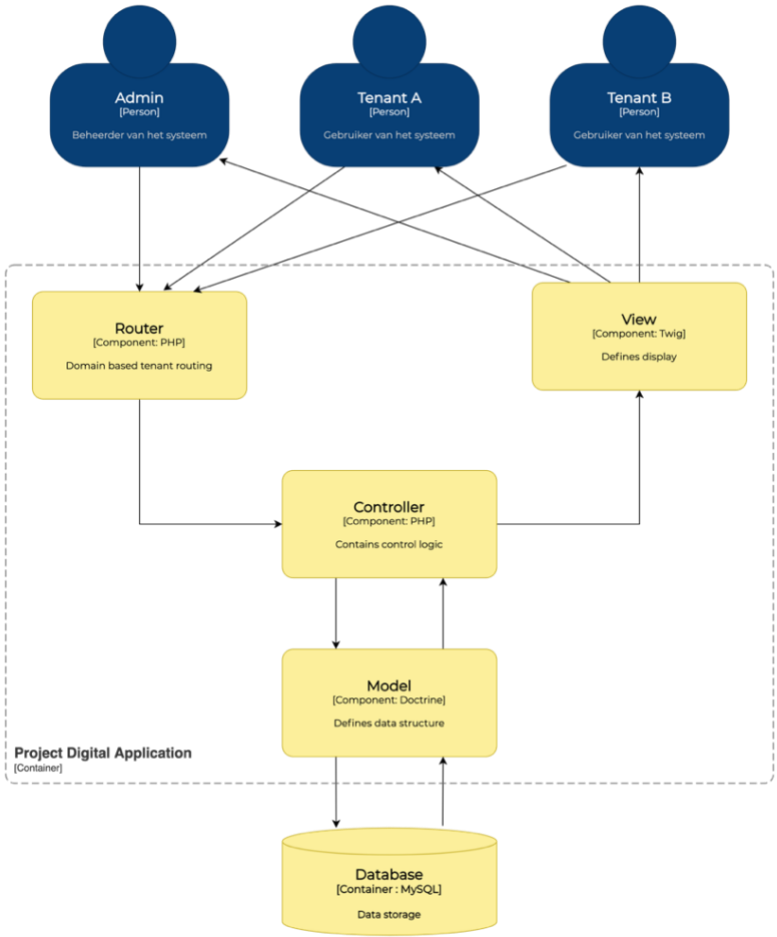
Het gebruik van aangepaste schema's vergroot de complexiteit van de oplossing, vooral omdat het schema moet worden gedefinieerd voordat de database gebruikt kan worden. Het is moeilijk om een schema te wijzigen nadat er gegevens aan een tabel zijn toegevoegd.



Database extensies - (Betts et al., 2013)

Project Digital

In dit hoofdstuk wordt de huidige staat van Project Digital in kaart gebracht. Onderstaande figuur bevat een grove weergave van de architectuur Project Digital.



De structuur is relatief simpel: Het systeem is monolitisch gebouwd en bestaat dus uit één geheel, zonder dat er gebruik wordt gemaakt van microservices of andere randsystemen/services. Aan de applicatie zit een SQL database verbonden waarin de data opgeslagen wordt.

Het project is gebouwd op het PHP Symfony framework en is opgebouwd volgens het MVC (Model-View-Controller) patroon. Dit patroon specificeert dat de applicatie in drie delen wordt opgedeeld, met elk een eigen functie. De model (of entiteit) bevat de data structuur en de business logic, de view bevat de weergave/user interface, de controller bevat alle control logic waarmee het geheel wordt georchistreerd.

De applicatie is ontwikkeld met een SaaS als uitgangspunt volgens het pool model. Zowel de admin als alle tenants maken gebruik van dezelfde omgeving en delen daarom alle resources. Alle data wordt in dezelfde database opgeslagen in gedeelde tabellen met aparte keys.

Identity and Access Management (IAM)

De implementatie van security en authenticatie wordt in het huidige systeem bereikt met behulp van Symfony Bundles. Deze bundles zijn als het ware plugins die kant-en-klare functionaliteit bevatten en dienen als een interne identity provider. De geïnstalleerde paketten zijn first-party bundles zijn die 'core'-functionaliteit van Symfony bevatten.

Omdat alle gebruikers van het systeem in één pool zitten, brengt dit bepaalde risico's met zich mee. Zo zou een gebruiker via bepaalde route zou toegang kunnen krijgen tot bepaalde plekken in de applicatie, waar dit niet zou mogen. Om dit te voorkomen zijn er 'voters' geimplementeerd.

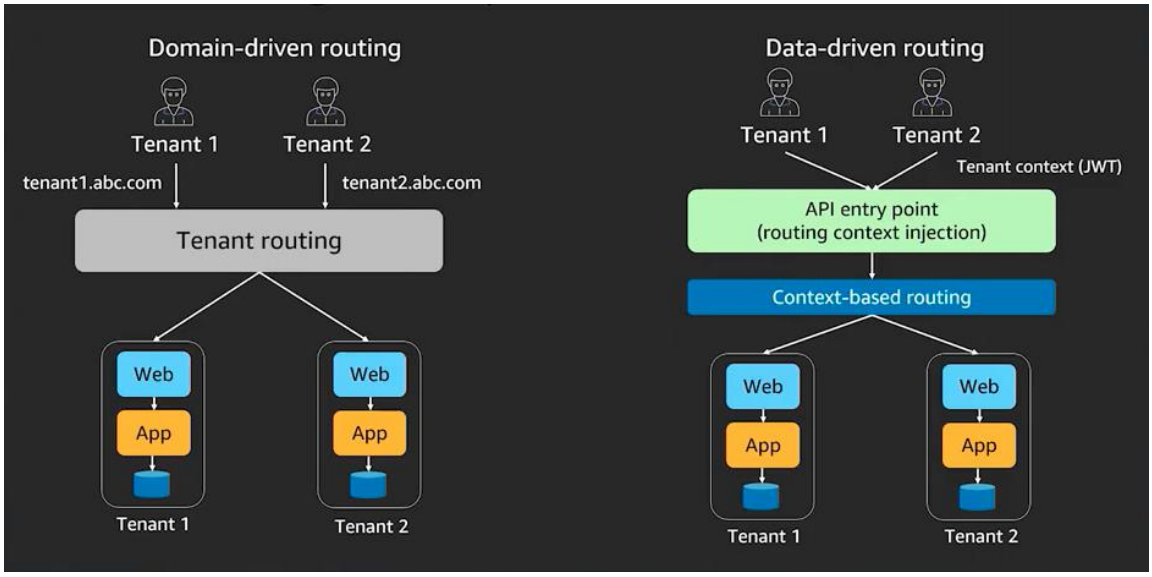
Voters zijn in principe hetzelfde als runtime policies. Voters bevatten een handmatig ingestelde lijst met toegangsvereisten. De vereisten kunnen bijvoorbeeld te maken hebben met de rol of identiteit van een gebruiker. Wanneer een gebruiker toegang vraagt tot een bepaalde resource wordt er door de voters 'gestemd' op elk van de vereisten. Toegang wordt geweigerd als er niet aan alle vereisten voldaan wordt. Bijzonder is wel dat voters vooral op gebruikersniveau werken en dat er dus minder focus ligt op de scheiding van tenants en juist meer op de splitsing van gebruikers.

Provisioning

Op het gebied van provisioning/deployment hoeft is er weinig nodig, omdat nieuwe tenants worden toegevoegd aan een bestaand systeem. Er hoeft dus geen nieuwe infrastructuur aangelegd te worden, alleen de routing hoeft geconfigureerd te worden (zie volgende paragraaf). Momenteel is dit nog niet volledig geautomatiseerd: Nieuwe tenants en de daaronder vallende gebruikers moeten handmatig aangemaakt worden door de administrator in de admin panel van de applicatie.

Routing

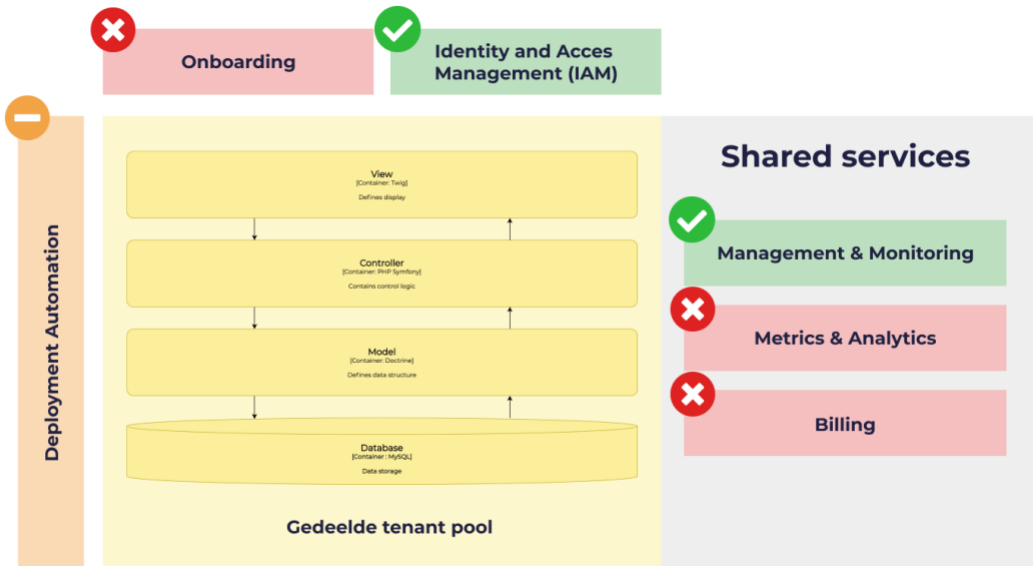
De applicatie verwijst tenants door naar de juiste omgeving door middel van domain-based routing. Elke tenant heeft hierbij een eigen subdomein (zie linkervoorbeeld in de onderstaande figuur). Door middel van een SSL Wildcard Certificate zijn alle subdomeinen automatisch over een HTTPS verbinding bereikbaar.



Tenant routing technieken - (Golding, 2020)

Conclusie

Onderstaande figuur geeft de huidige status van Project Digital als SaaS-oplossing weer. Vooral de implementatie van randservices zijn nog nodig.



Project Digital SaaS architectuur – Notitie: Adaptatie van 'SaaS architecture' diagram (Golding, 2020)

Op het gebied van architectuur is er al rekening gehouden met meerdere tenants en gebruikers, het pool model waarop de structuur is gebaseerd, verleent zich prima voor een SaaS applicatie.

Op het gebied van Identity & Acces Management zijn er maatregelen genomen om ongeautoriseerde toegang te voorkomen. Omdat deze handmatig zijn aangelegd is het wel belangrijk om deze periodiek te controleren en herzien. Ook zijn frequente backups van de database geen slecht idee, omdat meerdere gebruikers erin zitten en er dus mogelijk data van veel gebruikers verloren kan gaan.

Ook op management en monitoring vlak is Project Digital voorzien. Alle tenants bevinden zich in hetzelfde systeem, dus de reguliere monitoring van de server is in dit geval genoeg. Ook is er al een admin panel gebouwd om tenants te beheren.

Deployment automation is bijna compleet. De functies om nieuwe tenants en gebruikers aan te maken bestaan al, maar deze moeten wel beide vanuit een centraal punt aangeroepen kunnen worden wanneer een nieuwe tenant zich aanmeld.

Op korte termijn moet er een koppeling tot stand gezet worden met een billing provider en op langere termijn moeten metrics en analytics geïmplementeerd worden, om de status en het verbruik van individuele tenants te kunnen meten.

Als laatste moet er een onboarding mechanisme worden ontwikkeld, die de provisioning en de koppeling met het billing systeem in werking zet.

De implementatie van de cruciale SaaS functies (onboarding, deployment automation en billing) zal ongeveer 24 uur kosten (= 1 development sprint). De implementatie van metrics en analyics is erg afhankelijk van de complexiteit en kost 2 tot 3 sprints.

Discussie

- De onderzoeksresultaten zijn tot stand gekomen uit persoonlijk onderzoek en mogelijke vooringenomenheid van deze onderzoeker.
- De resultaten zijn afhankelijk van de zoekmachineresultaten, welke kunnen zijn beïnvloed door het zoekalgoritme van de desbetreffende zoekmachines en de zoekgeschiedenis van het gebruikte apparaat (cookies). Ook zijn de resultaten sterk afhankelijk van de gebruikte zoektermen en kunnen andere zoektermen hebben geleid tot een compleet ander resultaat.

Bronnen

- Abdul, A. O., Bass, J., Ghavimi, H., MacRae, N., & Adam, P. (2018). Multi-tenancy Design Patterns in SaaS Applications: A Performance Evaluation Case Study. *International Journal for Digital Society*, *9*(1), 1367–1375. https://doi.org/10.20533/ijds.2040.2570.2018.0168

- Amazon Web Services, Inc., & Golding, T. (2020). *SaaS Tenant Isolation Strategies*. https://d1.awsstatic.com/whitepapers/saas-tenant-isolation-strategies.pdf

- Auth0® Inc. (z.d.). JWT.IO - JSON Web Tokens Introduction. JSON Web Tokens - Jwt.Io. Geraadpleegd op 2 januari 2022, van https://jwt.io/introduction

- Betts, D., Homer, A., Jezierski, A., Narumoto, M., Zhang, H., & Hilf, B. (2013). *Developing Multi-tenant Applications for the Cloud on Windows Azure* (3de ed.). Microsoft Corporation.

- Chong, F., Carraro, G., Wolter, R., & Microsoft Corporation. (2006). Multi-Tenant Data Architecture. . Published. https://ramblingsofraju.com/wp-content/uploads/2016/08/Multi-Tenant-Data-Architecture.pdf

- Golding, T. [AWS Events]. (2017, 29 november). *AWS re:Invent 2017: GPS: SaaS Metrics: The Ultimate View of Tenant Consumption (GPSTEC308)* [Video]. YouTube. https://www.youtube.com/watch?v=X2SgoAl1vK8

- Golding, T. [AWS Events]. (2018, 30 november). *AWS re:Invent 2018: Deconstructing SaaS: Building Multi-Tenant Solutions on AWS (ARC418-R1)* [Video]. YouTube. https://www.youtube.com/watch?v=mwQ5lipGTBI

- Golding, T. [AWS Events]. (2019, 5 december). *AWS re:Invent 2019: SaaS tenant isolation patterns (ARC372-P)* [Video]. YouTube. https://www.youtube.com/watch?v=fuDZq-EspNA

- Golding, T. [AWS Events]. (2020, 24 juli). *SaaS Migration: Moving From Single-Tenant to Multi-Tenant on AWS - Level 300 (United States)* [Video]. YouTube. https://www.youtube.com/watch?v=Zqvt75TSKRA

- Golding, T. [AWS Events]. (2021, 5 februari). *AWS re:Invent 2020: Monolith to serverless SaaS: Migrating to multi-tenant architecture* [Video]. YouTube. https://www.youtube.com/watch?v=oso7Pb33LK8

- HAN University of Applied Sciences. (z.d.). Literature Study. CMD Methods Pack. Geraadpleegd op 20 september 2021, van https://www.cmdmethods.nl/cards/library/literature-study

- Krebs, R., Momm, C., Kounev, S., & Karlsruhe Institute of Technology. (2015). Architectural Concerns in Multi-Tenant SaaS Applications. . Published. https://se2.informatik.uni-wuerzburg.de/pa/uploads/papers/paper-371.pdf

- York, J. (2017, 2 september). *The SaaS Hybrid Question : Demystifying Software Business Models*. Chaotic Flow | Streamlined Angles on Turbulent Technologies by Joel York. Geraadpleegd op 20 oktober 2021, van https://chaotic-flow.com/the-saas-hybrid-question-demystifying-business-models/