NTNU
Kunnskap for en bedre verden

BACHELOROPPGAVE:

STYRINGSVERKTØY FOR NETWORK AS A SERVICE

FORFATTERE:
Bjørn Ole Myrold
Sondre Ahlgren
Snorre Rogne
Guro Storlien

Dato: 16.05.2018

Sammendrag

Tittel:	Styringsverktøy for Network As A Service	16. mai 2018
Deltakere:	Bjørn Ole Myrold Sondre Ahlgren Snorre Rogne Guro Storlien	
Veileder:	Eigil Obrestad	
Oppdragsgiver:	Evry Norge AS	
Stikkord:	NaaS, Aruba Central, REST-API, Database, Oracle, PL/SQL	
Antall sider: 192	Antall vedlegg: 9	Tilgjengelighet: åpen

Sammendrag:

Network as a Service (NaaS) er et konsept som EVRY Norge AS tar i bruk, der de leverer en tjeneste som involverer å sette opp, administrere og overvåke nettverk til deres kunder. Gjennom produsenten Aruba og deres skybaserte verktøy Aruba Central, kan EVRY tilby blant annet aksesspunkter og switcher uten at kunden trenger å eie eller forvalte eget utstyr eller teknologi. Men, EVRY vil ha en applikasjon med ekstra funksjonalitet, som kan hjelpe dem i denne prosessen.

Hensikten med *styringsverktøyet* er derfor å kunne se kontinuerlig oppdatert informasjon om kunder med tilhørende enheter og nettverk, på en storskjerm som er plassert i oppdragsgivers lokaler. For å vise informasjon, henter systemet data gjennom Aruba Central sitt REST-API og lagrer informasjon i en database. *Styringsverktøyet* fremstiller denne informasjonen i en webapplikasjon.

Abstract

Title:	Styringsverktøy for Network As A Service	16. mai 2018
Participants:	Bjørn Ole Myrold Sondre Ahlgren Snorre Rogne Guro Storlien	
Supervisor:	Eigil Obrestad	
Employer:	Evry Norge AS	
Keywords:	NaaS, Aruba Central, REST-API, Database, Oracle, PL/SQL	
Number of pages: 192	Number of appendixes: 9	Availability: open

Summary:

Network as a Service (NaaS) is a concept used by EVRY Norge AS, through which they provide a service that involves setup, administration, and monitoring of their customers networks. Through the producer Aruba and their cloud-based solution Aruba Central, EVRY can offer, among other tings, access points and switches without the need for the customers to own or manage their own set of equipment or technology. But, EVRY wants an application with added funcionality, that could help them in this process.

The purpose of *styringsverktøyet* is therefore to continuously display updated information about customers, with their afiliated units and networks on a big screen in EVRY's offices. To display this, the system retrieves the information through Aruba Centrals REST-API and stores this in a database. *Styringsverktøyet* displays the information in a web application.

Styringsverktøy for Network As A Service

BIDAT39 - Bacheloroppgave Dataingeniør

Bjørn Ole Myrold
Sondre Ahlgren
Snorre Rogne
Guro Storlien

Institutt for datateknologi og informatikk NTNU
Gjøvik, 16. mai 2018

Forord

Vi ønsker å rette en stor takk til EVRY Norge AS for tilliten dere har gitt oss, og for tett oppfølging undervegs i denne bacheloroppgaven. EVRY er et ledende nordisk IT-tjeneste og programvareselskap med mer enn 10 000 kunder i privat og offentlig sektor.

Denne oppgaven appellerte til oss, spesielt fordi den hadde et definert mål, men også på grunn av flere utfordringer på dens vei. Å være med på å utvikle noe som aldri har vært gjort før, har vært meget lærerikt.

Vi har hatt stor fornøyelse av å jobbe med dere og vi har opplevd å være en del av deres konsept og visjon «Embrace infinite opportunities». En spesiell takk til vårt *Orakel*, du har vært enestående.

Takk til vår veileder Eigil som har stupt ut i utfordringene sammen med oss, og sist men ikke minst venner og familie som har lest korrektur og kommet med tilbakemeldinger.

Innhold

Figurer

Tabeller

Definisjoner

NaaS-avtale Kunder som signerer en NaaS-avtale med oppdragsgiver kan bli lagt til i det utviklede systemet, og informasjon kan videre hentes ut gjennom Aruba Centrals API og vises i webapplikasjonen. 53

access token Akkreditiv for tilgang til API. ix, 33, 45, 62, 63, 78

akkreditiv Erstatningsord for det engelske ordet «credentials»: Nøkkelverdier som kreves for å få tilgang til overvåkingssystemet/informasjon/andre ressurser. 23, 34, 45, 52, 66, H.18

Aruba Bedriftsselskap under Hewlett-Packard (HP) som leverer nettverksløsninger og som er produsent i denne sammenhengen. Hjemmeside. viii, ix, 1, 7, 8, 20, 45, 46, 72, C.1–C.4, H.17, H.21, H.22

Aruba Central Aruba sitt sentrale overvåkningssystem som viser oversikt over forskjellige kunders nettverk, hvor mange enheter som er tilkoblet, og annen relevant informasjon angående nettverkstrafikk. viii, 1–3, 5, 16, 18, 26, 28, 32, 44, 45, 47, 53, 62, 63, 66, C.1–C.4, H.17, H.18, H.21

Aruba TAC Aruba's «Technical Assistance Center». Har bistått utviklerne med spørsmål vedrørende API-ets struktur og virkemåte. 20, 73, 78

endpoint Her brukt om et inngangspunkt, for eksempel en URL, mot en IT-service. En slik service kan for eksempel bli tilbudt av Aruba. 45, 46

enhet I denne rapporten definert som aksesspunkt eller switch, om ikke annet er spesifisert. viii, 3, 18, 30, 33, 37, 51, 56, 57, 60, 77, 82, C.2

EVRY Norge AS Et av Norges største IT-firma. Leverandør av IT-tjenester. Hjemmeside. viii, ix, 1–3, 7, 8, 18, 72, C.1, F.1, H.3, H.16–H.18, H.25

Managed Service Provider I denne rapporten oftest brukt i sammenheng med MSP-bruker: En «hovedbruker» som har rettigheter over kunder tilknyttet til den. Som en leverandør av Aruba Central, har EVRY Norge AS fått tilgang til en MSP-bruker for å administrere og overvåke sine egne kunder gjennom Aruba Central. xi, 45

OAuth2 OAuth 2.0 er en protkoll for autentisering, og er industristandaren for dette. (Fritt etter hjemmesiden). 45

On-site customer En kunde (her oppdragsgiver) som er aktivt deltakende i et prosjekt. 11, 84, C.7

oppdragsgiver Oppdragsgiver for dette prosjektet, EVRY Norge AS. viii, ix, 1–3, 5–7, 9, 11, 13, 14, 17–22, 32, 33, 36, 43, 46, 47, 52, 53, 56, 61, 66, 68, 70, 72, 75, 79, 81, 82, 86, C.1–C.4, C.10, H.17, H.21

Oracle En bedrift med over 400 000 kunder i 175 land som leverer konkuransedyktige løsninger innenfor software as a service, platform as a service, infrastructure as a service, og data as a service. (Fritt etter hjemmesiden). x, 18, 19, 46, 57, 63, 64, 68, 75, H.18, H.21

Oracle Application Express Et rammeverk som lar deg designe, utvikle og lansere flotte, responsive, databasedrevne applikasjoner kun ved bruk av en nettleser.(Fritt etter hjemmesiden). x, xi, 18

overvåkingssystemet Et navn på det totale styringsverktøyet utviklerteamet lager for oppdragsgiver. Dette inkluderer en database, med prosedyrer og funksjoner for manipulering av denne, og tilhørende webapplikasjon. viii, 9, 42, 45, 51, 52, 54, 60, 63, 64, 66, 68, 84–86, H.14

overvåkningsteam Ansatte ved EVRY Norge AS som overvåker kundenes trådløse nettverk, og som i hovedsak skal ta i bruk løsningen som utvikles under dette prosjektet. Vil ofte omtales som «bruker/brukerne». 2–4, 14, 23, 25, 60, 68, 79, 82

PL/SQL Et prosedyrebasert programmeringsspråk designet som en «utvidelse» av SQL og skal gi optimal effektivitet for kombinerte kodesnutter av SQL og PL/SQL. Dette innebærer introduksjon av funksjoner og løkker til tradisjonell SQL.(Fritt etter hjemmesiden). x, 19, 20, 46, 54

postman Postmann er et utviklingverktøy for API. Hjemmeside. 73, H.18

problem management Prosessen som har ansvaret for livsløpet for alle problem som har eller kan skje i et IT-system. C.3

produsent Se Aruba. 1

refresh token Akkreditiv for å generere nye access token. 33, 45

scheduled job Funksjonalitet i Oracle Application Express som lar en utføre oppgaver ved faste tidsintervall. 62, 79, F.3

SQL Developer Oracle SQL Developer er et gratis, integrert utviklermiljø som forenkler utviking og administrering av Oracle-databaser. Verktøyet tilbyr editering og kjøring av PL/SQL og vanlige SQL-utsagn, konsoll for databaseadministrator (DBA) og en komplett datamodelleringsplattform. (Fritt etter hjemmesiden). 20

Taiga Et scrum-verktøy valgt av utviklerteamet som hjelp til å planlegge, estimere og gjennomføre prosjektet. 70, H.24, H.25

utviklerteamet Består av utviklerene: Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren (GSBOS). ix, x, 1–3, 5, 9, 11–13, 16, 18–21, 32, 47, 52, 63, 69, 75, 81, H.21

wallet En del av lagringsområde til en tjener som «hoster» en Oracle-database. Tjeneren lagrer akreditiver og sertifikater brukt av/i databasen i dette området. 46, 50, 75, 82

Forkortelser

APEX Oracle Application Express. 18, 19, 23, 42, 43, 52, 54, 71, 75, 79, H.18, H.21, *Glossary:* Oracle Application Express

GSBOS Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren. x, 1, 7, C.1, C.2, C.4, F.1

HP Hewlett-Packard. viii

IDI Institutt for Datateknologi og Informatikk. 8

MSP Managed Service Provider. 45, 50, 78, *Glossary:* Managed Service Provider

NaaS Network as a Service. viii, 1, 53, 72, C.1, H.17

SLA Service Level Agreement. 4, C.1, C.2

1 Innledning

Både teknologien og mennesket er i kontinuerlig endring, og teknologi blir en stadig større del av menneskers hverdag. Nye løsninger, tjenester og idéer blir satt på prøve hver eneste dag og nye behov blir skapt. Utviklingen av ulike teknologiske løsninger går så raskt fremover at utfordringer lett kan oppstå. Det har derfor blitt viktig å ha gode verktøy for å møte teknologiens eksponentielle vekst[1] og tilstrebe innovative løsninger. Skytjenester er veldig i vinden nettopp på grunn av at utfordringen med raskt endrende teknologi kan være enklere å håndtere dersom en skyløsning blir brukt[2]. Slike skytjenester håndterer alt fra dataprosessering og datalagring, til programvare på tjenere i eksterne tjenerparker tilknyttet internett[3].

Oppdragsgiver EVRY Norge AS tilbyr skytjenester, deriblant Network as a Service (NaaS), til sine kunder. Gjennom NaaS tilbyr de å sette opp, konfigurere og eventuelt drifte et eller flere nettverk for kundene, slik at kundene kan fokusere på sin spesifikke kompetanse innenfor de tjenestene de selv tilbyr[4]. I dag tilbyr EVRY Norge AS denne tjenesten som en komplett pakke der kunder kan leie og/eller kjøpe nettverksløsninger til sine kontorer, eller lignende. Da utføres en vurdering av kundenes behov for lokasjonene som skal dekkes av WiFi og/eller kablet nettverk, og setter opp dette ferdig konfigurert, klart til bruk.

Via tjeneste-produsenten Aruba, har EVRY Norge AS et overvåknings- og administrasjonsprogram, kalt Aruba Central, for nettverk. Gjennom dette programmet kan oppdragsgiver administrere og se nettverkene til kundene sine. Ved å benytte et REST-API mot Aruba Central sin løsning, ønsker oppdragsgiver at relevant informasjon skal hentes ut og vises på en oversiktlig måte. Det er dette bachelorgruppen(utviklerteamet), bestående av Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren (GSBOS) skal utføre.

1.1 Bakgrunn

I dag har mange bedrifter egne nettverksløsninger for seg selv og deres gjester, uten at de sitter på spesialkompetanse innenfor området. Bedrifter som

for eksempel kjøpesentere, hoteller og fylkeskommuner har gjerne komplekse løsninger som det er vanskelig å holde oversikt over. Eksempelvis vet nødvendigvis ikke hotellbransjen så mye om nettverkadministrasjon, de skal være gode på å drive et hotell. Etter prinsippet «rett person til rett jobb» kan de sette bort oppgaven å drifte nettverk til andre som har bedre kompetanse på området. EVRY Norge AS er et slikt firma.

Flere bedrifter vurderer nå muligheten for å ta i bruk tjenesten NaaS. Oppdragsgiver ser at Aruba Centrals løsning ikke fungerer så godt som de ønsker, spesielt med tanke på en eksponentiell vekst i fremtiden. Løsningen krever mye navigering i grensesnittet for å vise ønsket og relevant informasjon, den er derfor tungvinn og «informasjonsfattig». Dette gjør nåværende løsning lite ergonomisk og effektiv for de ansatte(Overvåkningsteamet) som jobber med Aruba Central. Oppdragsgiver trenger en løsning som kan overvåke nettverkene de drifter og som kan videreutvikles med funksjonalitet som for eksempel viser relevant informasjon, rapporter over tidligere situasjonsbilder, og kan lette hverdagen til overvåkningsteamet hos Evry.

1.2 Avgrensning

Oppdragsgiver har kommet med en overordnet problemstilling(se vedlegg B), og utviklerteamet har dermed fått mye frihet til å selv bestemme hva som skal inngå i den ferdige løsningen. Hovedfokuset er å bygge en god grunnmur, ved å modellere og sette opp en robust database som enkelt kan benyttes og skaleres. Dette er med på å sikre gode muligheter for videreutvikling, økt funksjonalitet og merverdi for oppdragsgiver.

Aruba Central har store mengder innsamlede data om de ulike nettverkene, men løsningen som utvikles i dette prosjektet skal kun hente ut og lagre de data som oppdragsgiver har vurdert som relevant å vise. Oppdragsgiver ønsker at løsningen skal bestå av en webapplikasjon med få sider. Videre skal en innloggingside implementeres for å forhindre utilsiktet tilgang til applikasjonen. I tillegg er det ønskelig at innholdet på hver enkelt side skal være begrenset til det som er absolutt nødvendig. Utviklerteamet har bestemt seg for å begrense mengden av grafiske fremstillinger, og vil heller ha fokus på en enkel, informativ og lettleselig fremstilling i form av en tabelloversikt.

1.3 Oppgavedefinisjon

Det blir utviklerteamets oppgave å lage en oversiktlig webapplikasjon som krever lite eller ingen navigering i grensesnittet for å vise informasjon over nettverkene med tilhørende enheter. Oversikten skal bare hente informasjon om de kundene oppdragsgiver drifter og yter suppert til. Styringsverktøyet skal med andre ord fremstille et komplett, og oversiktiktlig grensesnitt, som gir relevant informasjon med tanke på drift og overvåkning av nettverkenes status og omfang. Løsingen skal presenteres og brukes i det daglige på en storskjermvisning tilpasset overvåkningsteam.

Databasen som webapplikasjonen skal bygges på, skal inneholde kontinuerlig oppdatert informasjon hentet fra Aruba Centrals API ved hjelp av HTTP-protokollens GET- og POST-metoder. Dette utføres ved å implementere prosedyrer som ved faste intervall utfører nye spørring mot API-et. Det skal også være mulig for en bruker å legge til kunder i databasen, gitt at denne spesifikke kunden har en eksisterende -avtale med EVRY Norge AS. Kunder som signerer på en slik avtale, blir lagt til i systemet, som så henter informasjon om alle dens nettverk og enheter gjennom API-et.

1.4 Prosjektmål

Hensikten med denne oppgaven er å gjøre arbeidet med å overvåke og yte support for kundene mer effektivt for overvåkningspersonellet. Løsningen skal være med på å sikre god og riktig kvalitet på tjenesteleveransen EVRY Norge AS utfører.

1.4.1 Resultatmål

Resultatet skal være en fungerende webapplikasjon som viser verdifull informasjon til bruker. Dette innebærer:

- Oversiktsbilde på storskjerm over oppdragsgivers kunder, med tilhørende nettverk og enheter, og oppdatert status på disse.
- En ergonomisk løsning som krever lite navigering for overvåkningsteam.

- Sortering etter alvorlighetsgrad med terskler definert av overvåkningsteam(for eksempel en enhet er ute av drift)
- Et sikkert system, blant annet i henhold til kommende GDPR-bestemmelser og oppdragsgivers øvrige krav.

1.4.2 Effektmål

Overvåkningspersonellet skal få en bedre oversikt over de ulike kundene, deres nettverk og på den måten yte mer effektivt support. Dette skal skape merverdi for oppdragsgiver som leverandør.

- Senke responstid ved feilmeldinger.
- Øke tilgjengeligheten på informasjon fra WiFi-nettverk.
- Enklere overholdelse av Service Level Agreement (SLA) med den enkelte kunde.
- Øke kundetilfredshet.

1.4.3 Læringsmål

Etter gjennomført bacheloroppgave skal bachelorgruppen ha tilegnet seg kunnskap, ferdigheter og generell kompetanse tilknyttet dataingeniøryrket. Teamet vil ha fokus på å tilegne seg kunnskap og ferdigheter innenfor områdene; problemstilling, identifisere og vurdere relevant litteratur, og utarbeide konkrete løsningsalternativer til problemstillingen. Gruppen vil også øke sin generelle kompetanse innenfor vitenskapelig redelighet og forståelse for etiske problemstillinger som er av relevans for problemstillingen. Mer informasjon om læringsmål finnes på NTNUs informasjonsider[1].

Gruppen ønsker å tilegne seg mest mulig erfaring innen utvikling av en større applikasjon, gjerne ved hjelp av moderne og bransjerelaterte verktøy og programmeringsspråk. Ett tett samarbeid mellom medlemmene vil også sikre et læringsrikt miljø der deling av kunnskap innad i teamet skal stå sterkt.

[1] https://www.ntnu.no/studier/emner/BIDAT39#tab=omEmnet

1.5 Rammer

Oppdragsgiver beskrev at formålet med denne oppgaven var å gjøre overvåkningen av nettverkene mer effektiv, enn slik den var ved prosjektets start. Hvordan teamet skulle løse denne problemstillingen, ønsket oppdragsgiver å «sparre» med teamet for å komme til enighet om. Prosjektavtalen ble signert i januar 2018 og markerte starten av prosjektet. Tidsrammen var fra dette tidspunktet til og med prosjektets slutt i mai. Kildekode laget for og under prosjektet er eiendeler av oppdragsgiver. Dette står nærmere beskrevet i prosjektavtalen (vedlegg A).

1.5.1 Teknologi

Løsningen vil komme i form av en webapplikasjon og vil dermed måtte bli «hostet» på en server. Oppdragsgiver har i utgangspunktet få preferanser for hvilken type teknologi teamet skal utarbeide løsningen i.

Løsningen er avhengig av å kommunisere med Aruba Centrals API. Oppdragsgiver har prøvd å få tilgang til API-et i et tidligere prosjekt, men hadde verken tid eller ressurser til å finne ut hvordan tilkoblingen fungerte. Dette er potensielt et stort problem, da denne kommunikasjonen er essensiell for å utvikle dette produktet. Det er uvisst hvor lang tid det vil ta for utviklerne å løse dette problemet, og om det i det hele tatt er mulig.

Om kommunikasjonen kommer på plass, er det heller ikke sikkert at det er mulig å hente den informasjonen oppdragsgiver er ute etter gjennom API-et. Dette er derfor den største begrensende faktoren, og er beskrevet nærmere i ROS-analysen.

1.5.2 Gjennomføring

Utviklerteamet vil jobbe målrettet sammen for å nå målene som er satt. Noen tidsfrister som spesielt bør nevnes er:

- Start tirsdag, 09.01.18.
- Intern deadline på utvikling, 27.04.18.

- Levering av rapport, 16.05.18.
- Presentasjon, 04/05.06.18.

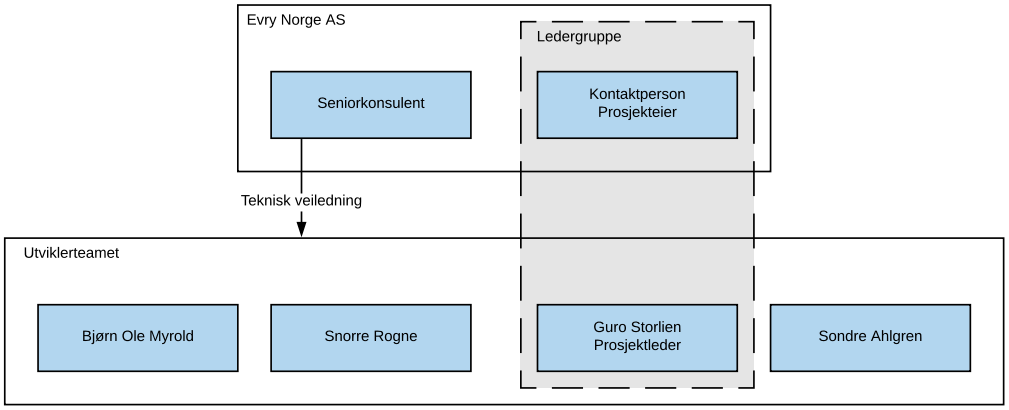
1.6 Egen bakgrunn og kompetanse

Samtlige gruppemedlemmer i dette prosjektet har i stor grad samme akademiske bakgrunn. Alle studerer dataingeniør ved NTNU i Gjøvik, og har jobbet sammen i noen felles prosjekter gjennom studiet, hvor et av dem var et større prosjekt i emnet *systemutvikling*. I dette emnet fikk gruppemedlemene god innføring i ulike systemutviklingsmodeller og anvendelsen av disse. Andre emner som gruppen har fått god nytte av, er; *applikasjonsutvikling*, med et større prosjekt programmert i Java, med tilnærmet samme omfang som løsningen som nå skal utvikles; og *datamodellering og databasesystemer*, hvor gruppen fikk en god introduksjon til hvordan korrekt modellere og sette opp en SQL-database.

Vitenskapelig programmering har vært et valgfag som flere av utviklerne har dratt nytte av, da det blant annet har tatt for seg prosjekter med en vitenskapelig tilnærming. Guro og Snorre har i tillegg hatt emnet objektorientert systemutvikling, hvor detaljert design av løsninger, arkitektur, designmønstre og bruk av ulike rammeverk er meget relevant. Videre har Sondre hatt *programmerbar infrastruktur* og Bjørn Ole hatt *programvaresikkerhet*.

1.7 Prosjektorganisering

Gruppen fikk tildelt et fast rom i oppdragsgivers lokaler hver tirsdag og torsdag. Dette var meget gunstig da det muliggjorde hyppig kommunikasjon med oppdragsgiver under utviklingsperioden. På disse dagene kunne gode beslutninger tas fortløpende, spørsmål kunne raskt besvares og teamet kunne jobbe effektivt sammen. De øvrige dagene skal teamet tilstrebe å jobbe sammen i kjernetiden 08-16. Flere grupperegler er beskrevet i vedlegg D. Teamet besluttet at Guro skulle være teamleder, særlig på grunn av hennes ledererfaring som befal på rekruttskole. Grunnet gruppens størrelse, har hun også rollen som utvikler i teamet. Organisasjonsstrukturen ble da som vist på figur 1.1.



Figur 1.1: Organisasjonskart for prosjektet

Kontaktperson representerer EVRY Norge AS i denne sammenheng, og er prosjektets eier.

Seniorkonsulent har mye kunnskap om Aruba sine systemer. Konsulenten bistår i tekniske spørsmål.

Guro Storlien er prosjektleder og leder for utviklerteamet, som består av Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren (GSBOS).

Hovedkommunikasjonen mellom utviklerteamet og oppdragsgiver vil foregå mellom Kontaktperson og Guro Storlien. Det vil også være andre ressurspersoner hos oppdragsgiver som bistår utviklingen med veiledning, testing og tilbakemeldinger.

Ulike arbeidsoppgaver delegeres undervegs i prosjektet etter demokratisk avgjørelse internt i utviklerteamet. Det blir med andre ord ikke forhåndsbestemt hovedansvar for oppgaver, dette på bakgrunn av ønske om å fordele læringsutbytte relativt likt mellom medlemmene i teamet. Utenom dette vil Eigil Obrestad, utnevnt av NTNU i Gjøvik, være veileder for teamet gjennom hele prosjektperioden.

1.8 Målgruppe

Rapport Hovedpublikum for denne rapporten er den akademiske omgangskretsen innen Institutt for Datateknologi og Informatikk (IDI) ved NTNU. De som eventuelt vil videreutvikle løsningen, i tillegg til andre som skal utvikle lignende løsninger mot produsenten Aruba vil også ha nytte av å lese rapporten.

Webapplikasjon Målgruppen er EVRY Norge AS sine ansatte som jobber med overvåking og support av kundenes nettverk, og er ute etter å effektivisere arbeidshverdagen sin.

1.9 Rapportstruktur

I denne rapporten skal språket tilpasses medstudenter innenfor IDI, og vil anta en viss kunnskap om velkjente «ord og uttrykk».

Rapporten vil bli skrevet i skriveverktøyet ShareLaTeX[1] som er et verktøy for skriving av alle typer dokumenter der man «koder» dokumentet. LaTeX har åpen kildekode og har dermed mange tilleggspakker utviklet til seg, der flere skal brukes i denne rapporten. Noen av disse tillater dynamiske kryssreferanser og linker i teksten, som vil være «klikk-bare» for en leser av PDF-versjonen.

Et par punkter å legge merke til i dette dokumentet som er gjort ved hjelp av ovennevnte verktøy:

Linker under en viss lengde skal formateres slik som denne: https://www.google.com. «Lengre» linker vil bli lagt inn under en tekst, som dette.

Titler defineres som kapittel, seksjon, underseksjon og paragraf; der kapittel er det øverste nivået og paragraf det nederste. Alle titler ned til underseksjon skal vise i innholdsfortegnelsen, unttatt for vedlegg, der bare kapittel vises.

[1] www.sharelatex.com. ShareLaTeX er en utvidelse av LaTeX, et system for å komponere tekst. De komponerte filene blir så kompilert til PDF av programmer for LaTeX, som for eksempel ShareLaTeX.

Kryss-referanser til andre deler av dokumentet og/eller en side vil ha en gråfarge, slik som referansen 1.9, som viser tittelnummeret til seksjonen Rapportstruktur. Dette gjelder også tall og/eller tekster som markerer fotnoter, figurer, tabeller, kodelistinger, definisjoner og forkortelser.

Definisjoner og Forkortelser er definert av utviklerteamet og dukker opp med jevne mellomrom i teksten. Disse fungerer som kryssreferanser tilbake til sin egen forklaring etter innholdsfortegnelsen, og vises altså i en nyanse av grå. Dette skal markere disse, men kan ikke brukes «over alt» da dette går utover lesbarheten i enkelte deler av dokumentet. Det er derfor bare de markerte ordene som er «klikk-bare». De fleste gjentagelser av ordet i en gitt sammenhengen(for eksempel en seksjon) referer til den samme definisjonen eller forkortelsen.

Rapporten skal ha følgende kapittel, her gitt med en overordnet forklaring av innholdet i disse:

Innledning Inneholder introduksjon til oppgaven og rapporten, samt ordlyden for oppgaven, litt om forfatterene og deres akademiske bakgrunn, organisering av prosjektet, og til slutt, struktur på rapporten.

Arbeidsmetode beskriver hvordan teamet planlegger å jobbe med bacheloroppgaven og hvilke rammer utviklingsprosjektet har.

Kravspesifikasjon har som formål å vise hvilke krav som er satt til overvåkingssystemet av oppdragsgiver og utviklerteamet, use case og ett overordnet sekvensdiagram for disse.

Design Her diskuteres designarkitekturer, hvilken som er valgt for systemet og det blir vist en «mockup» av et tenkt grafisk grensesnitt.

Implementasjon og Realisering er hoveddelen i denne rapporten og tar for seg store deler av utviklingsprosessen. Her går man også nærmere på teknologier, gir brukseksempler og forklarer i detalj hvordan spesifiserte krav er løst.

Testing og kvalitetssikring omhandler ulike tiltak for å teste og sikre kvaliteten til produktet og prosessen.

Utførelse inneholder sammendrag fra hver enkel periode, som oftest sprinter, i prosjektet, og de erfaringer som gruppen har opparbeidet seg.

Avslutning presenterer resultatet av prosjektet og konklusjonen basert på gjennomføring. Omfattter også en seksjon om tanker teamet har gjort seg i løp av prosjektets gang, spesielt hvordan systemet kan forbedres.

Vedlegg vil ha alle vedlegg tilhørende rapporten. Her kan man finne blant annet avtaler, logger og brukermanual.

2 Arbeidsmetode

Dette kapittelet beskriver hvordan teamet planlegger å jobbe med bacheloroppgaven og utviklingsprosjektet.

2.1 Valg av utviklingsmodell

Utviklerteamet skal utvikle en løsning over en periode på omtrent fire måneder. Teamet har derfor en stram tidsplan. Ved valg av utviklingsmodell drøftet gruppen flere gode argumenter for å velge en smidig tilnærming.

- Oppdragsgiver er en pådriver for bruk av smidig utvikling.
- Utviklerne har tidligere god erfaring med smidig utvikling, og foretrekker smidig fremfor plandreven.
- Prosjektet har ingen definert sluttilstand, da man ønsker å utvikle dette prosjektet basert på løpende «dynamiske» krav.
- En utfordring i prosjektet er å tidsestimere oppgaver, og vil derfor passe en smidig utvikling.
- Oppdragsgiver har ikke definert konkrete krav til ferdig løsning. Med smidig utvikling blir det enklere å komme med nye krav undervegs.
- Prosjektet har fire utviklere i teamet, dette passer også godt overens med størrelsen på et utviklingsteam i smidig utvikling.

Teamet drøftet følgende rundt noen kjente utviklingsmodeller:

Kanban er en relativt enkel modell som baserer seg på en del relevante prinsipper. Blant annet baserer den seg på åpenhet rundt diskusjoner og avgjørelser, tildeling av ansvar ned til enkeltpersoner og god kommunikasjon innad i teamet. Dette er noe som er forenelig med gruppens teamstørrelse på fire, og er prinsipper som vil bringes videre med i utviklingsarbeidet.

XP har også flere gode prinsipper som passer et utviklerteam på fire. On-site customer, som vil si tett kontakt med oppdragsgiver, vil være viktig

i dette prosjektet, da på grunn av at gruppen hver tirsdag og torsdag har tilgang på kontorplass hos oppdragsgiver. Dette vil tillate en tett dialog med oppdragsgiver undervegs, men må balanserens og gjøres via riktige kanaler.

Modellen implementerer parprogrammering som prinsipp, som vil være relevant å ta i bruk hvor for eksempel kodens kvalitet er viktig, eller utviklingen er spesielt krevende. XP tar også for seg en balansert arbeidshverdag hvor «skippertak» og mye overtid sjeldent skal forekomme. Dette er også prinsipper som teamet vil ta i bruk, og som er vurdert viktig for en god gjennomføring av bacheloroppgaven.

Scrum tilbyr en stor plattform med mange verktøy til smidig utvikling. Modellen beskriver en gitt struktur med sprinter som passer oppgavens art. Oppdragsgiver har presentert få punkter til kravspesifikasjon initielt, men vil komme med flere innspill undervegs. Iterativ utviklingsmodell passer godt til dette prosjektet, siden det er nødvendig å legge til nye arbeidsoppgaver undervegs. Det er også ønskelig med en utviklingsmodell basert på Scrum, da dette passer teamet i form av arbeidsmetodikk, selvstendighet og beslutningsdyktighet.

Scrum vil være utviklingsmodellen som i hovedsak skal følges, men prinsipper fra Kanban og XP vil bli integrert. Utfordringen med Scrum vil være at teamet ikke har brukt denne modellen i en lignende oppgave før, ei heller har erfaring med denne typen utviklingsprosjekt i så stor skala. På grunn av dette vil det å estimere de ulike «product backlog items (PBI)» i forkant av hver sprint være en utfordrende oppgave.

2.2 Scrum i praksis

Teamet velger å dele utviklingsfasen i seks sprinter med intervall på 10 arbeidsdager. Noen unntak for dette er en planlagt påskeferie for utviklerteamet: Sprint 4 avsluttes fredag 23.03.18 (uke 12), og sprint 5 vil da starte tirsdag 03.04.18 (uke 14). Dette medfører at sprint 5 effektivt vil bli 9 arbeidsdager lang. Etter endt påskeferie vil fokuset gå gradvis fra utvikling til skriving av prosjektrapporten, og ved slutten av uke 17 (fullført sprint 6) var det planlagt å avslutte utvikling. Da er forhåpentligvis produktet ferdigstilt, noe som

gir litt i overkant av to uker til ferdigstilling av rapporten for innlevering. Utviklingsfasens 6 sprinter:

- Sprint 1, 29.01 - 08.02
- Sprint 2, 12.02 - 23.02
- Sprint 3, 26.02 - 09.03
- Sprint 4, 12.03 - 23.03
- Sprint 5, 03.04 - 13.04
- Sprint 6, 16.04 - 27.04

Ettersom oppgavene som skulle gjennomføres var vanskelige å estimere på grunn av utfordringene rundt API-et, hadde utviklerteamet en god dialog med oppdragsgiver undervegs, blant annet for å vurdere hva som skulle gjøres for hver sprint. Flere detaljer rundt fremdriftsplan og Gant-diagram finnes i prosjektplanen, vedlegg C.

Teamet skal avholde «Daily Scrum» møter på rundt ti minutter for å holde alle i teamet oppdatert på den helhetlige prosessen, hva som ble utført dagen før og hva som skal gjøres den dagen. Etter hver sprint skal det avholdes «Retrospective» møte for alle utviklere, og ved behov «Sprint Review» møte med oppdragsgiver.

2.2.1 Dokumentasjonskrav

Arbeidsprosess

Store deler av selve arbeidsprosessen som foretas skal dokumenteres, slik at det blir enklere å se tilbake på ting som er besluttet og utført gjennom en gitt sprint. Toggl[1] er benyttet av teamet for å føre timelister med beskrivelser om hva som har blitt gjort. Møtereferater og dagslogger skal føres i loggboken.

[1]Verktøy for sporing av tid. https://toggl.com/

Ferdigstilt produkt

Løsningen tas i bruk av overvåkningsteamet straks denne er klar. Det er ønskelig at webapplikasjonen også skal testes undervegs i utviklingen for å få tilbakemeldinger fra personellet som faktisk skal bruke løsningen. Detaljer rundt dette vil bli avtalt undervegs med oppdragsgiver.

Kildekode

All kildekode skal dokumenteres etter standard for det aktuelle programmeringsspråket som tas i bruk under utviklingsprosessen. For språk som har integrerte dokumentasjonsprogrammer skal dette alltid benyttes. Øvrig gjelder god struktur og sammenheng mellom kode og kommentarer etter en standard som teamet har blitt enige om på forhånd. For eksempel skal hver enkel fil inneholde en «header» med informasjon om den aktuelle filen, etter en standard anbefalt av oppdragsgiver.

Møter og veiledning

Alle møter skal dokumenteres. Referater ligger i vedlegg H. Prosjektleder har ansvar for at dette blir utført og følges opp. Dette er beskrevet nærmere i kapittel 6.

2.2.2 Risikoanalyse

Identifikasjon og analyse

Identifisering og analyse av ulike risikoer ved dette prosjektet er viktig for teamet, fordi det er ønskelig med høy grad av måloppnåelse og en profesjonell arbeidsprosess. Håndtering av risiko skal være en god kultur i teamet og tiltakene som er satt skal etterfølges. Det benyttes følgende standard for risikoanalysen, hvor hver grad har fått et tall som brukes i risikoberegningen: **Sannsynlighet**: Særdeles liten (1), Liten (2), Trolig (3), Stor (4), Særdeles stor (5).

Konsekvens: Trivielt (1), Mindre alvorlig (2), Alvorlig (3), Kritisk (4), Katastrofalt (5).

Produktet av disse gir så en indikator på risikograden til de ulike scenario. Disse tolkes på følgende måte:

1 - 4 Lav risiko/grønt nivå.

5 - 12 Middels risiko/gult nivå.

13 - 25 Høy risiko/rødt nivå.

Se tabell 2.1 for full oversikt over identifiserte risikoer.

#	Risiko	Sannsynlighe	Konsekvens	Risiko
1	Utviklerteamet får ikke etablert forbindelse med Aruba Centrals API innen rimelig tid	Stor	Kritisk	Høy
2	Tausehetsbelagte opplysninger kommer på avveie	Liten	Katastrofal	Middels
3	Prosjektet blir ikke fullført innen gitte tidsrammer	Trolig	Kritisk	Middels
4	Kildekode, dokumentasjon, rapport går tapt	Liten	Katastrofal	Middels
5	Sykdom eller hendelser innad i prosjektgruppen som fører til fravær over større tidsperioder	Stor	Alvorlig	Middels
6	Andre firmaer utvikler lignende løsninger	Liten	Alvorlig	Middels
7	Store endringer i kravspesifisering underveis i prosjektperioden	Stor	Mindre alvorlig	Middels
8	Ønsket funksjonalitet kan ikke implementeres fordi det ikke er teknisk mulig eller det blir for liten tid	Trolig	Mindre alvorlig	Middels

Tabell 2.1: Identifiserte risikoer på middels og høyt nivå

Plan for håntering av risiko

Tabell 2.2 viser en oversikt over et utvalg av de identifiserte risikoene i prosjektet, med tilhørende planlagte tiltak for disse.

#	Tiltak
1	Teamet skal tilstrebe å jobbe med verktøy og funksjonalitet rundt selve løsningen selv om kontakten med API-et ikke er på plass. I påvente av respons eller hjelp, skal teamet jobbe offensivt med andre muligheter. Preventivt vil teamet bruke tiden før utviklingen starter til å utforske relevante emner og tilegne seg kunnskap rundt tema.
2	Tausehetsbelagte opplysninger skal håndteres varsom og etter oppdragsgiver sine rettningslinjer.
3	Sørge for god planlegging og gjennomføring av prosjektet. Tett oppfølging av utfordringer undervegs. Ha fokus på dokumentasjon slik at eventuelle nye bachelorgrupper eller ansatte hos oppdragsgiver kan fortsette prosjektet senere.
4	Teamet skal følge gode rutiner for behandling, lagring og backup av kildekode, dokumentasjon og rapport.
5	Teamet oppfordres til sunn og god livsstil for å unngå sykdom og skader. For å minimere konsekvensen ved eventuelle tilfeller, skal alle utviklere presentere kort hva de har utført av arbeid under «Daily Scrum». Ved større eller komplisert funksjonalitet skal ekstra tid settes av til dette. God dokumentasjon i henhold til rutiner undervegs skal også utføres for å minimere tap av fremgang ved fravær.
7	Bruke prinsipper innenfor smidig utvikling og scrum. Med dette menes blant annet god kommunikasjon og forståelse, hyppige og produktive møter med oppdragsgiver hvor fremgang, status og videre arbeid står på agendaen.
8	Utfordringer ved ny ønsket funksjonalitet skal adresseres så raskt som mulig, oppdragsgiver skal bli gjort oppmerksom på dette, og eventuelle tiltak eller endringer blir utført fortløpende. En liste med funksjonalitet som kan videreutvikles, overleveres til oppdragsgiver ved prosjektets slutt.

Tabell 2.2: Risikotiltak

2.3 Rammeverk og verktøy

I dialog med oppdragsgiver har verktøy og rammeverk blitt valgt basert på utviklerteamets egne preferanser og vurderinger.

2.3.1 Valg av utviklingsmiljø

Initielt var det planlagt å utvikle en løsning i JavaScript med rammeverkene Node.js[1] og React[2]. Dette var før oppdragsgiver uttrykte ønsker om å kunne føre historikk over data, slik at dette igjen kunne presenteres for eventuelle kunder i form av grafer, diagram, eller lignende. Et slikt ønske stilte krav til at løsingen tar i bruk en database for å mellomlagre data hentet fra Aruba Centrals API. Det ble også påpekt av oppdragsgiver at de ville ha en robust database som kunne legge et godt grunnlag for videreutvikling av løsningen, som også håndterer en eksponentiell økning i antall kunder og enheter og nettverk under disse. For å muliggjøre disse ønskene måtte teamet vurdere hvordan en database skulle bli implementert mot Node.js og React.

Under denne prosessen ble det i et møte med en ansatt hos EVRY Norge AS, presentert platformen Oracle Application Express (APEX) som utviklerteamet kunne jobbe på. Dette verktøyet er brukt av oppdragsgiver internt, og de har god kompetanse på dette. Teamet stilte seg i begynnelsen skeptisk til denne plattformen, grunnet frykt for at verktøyene på plattformen ville løse for mange av utfordringene i prosjektet. I dialog med veileder ble det derimot poengtert at det er viktig å bruke de verktøy som er tilgjengelige, og som komplimenterer den oppgaven som skal utføres i størst mulig grad. På den måten trenger man ikke å «finne opp hjulet på nytt», og kan heller bruke tid på utvikling av andre deler av systemet.

Oracle

Etter en samtale med en annen ansatt hos EVRY Norge AS som er godt kjent med Oracle, deres forskjellige verktøy, og hvordan disse fungerer sammen,

[1] https://nodejs.org/en/
[2] https://reactjs.org/

bestemte utviklerteamet seg for å bruke APEX på bakgrunn av fordelene dette ga teamet. Plattformen er en del av Oracles databasedistrubisjoner[1] og tilbyr mye funksjonalitet, da i hovedsak muligheter for å lage grafiske brukergrensesnitt mot den nevnte databasen. I tillegg til at det er en utbredt plattform, er dette noe oppdragsgiver bruker internt i bedriften, og det blir da enklere for de å eventuelt videreutvikle løsningen i etterkant. Dette gjorde det også lettere for oppdragsgiver å sette opp tjeneren for løsningen. I tillegg ville APEX være et nytt verktøy for teamet, noe som gir utviklerene nyttig erfaring, samtidig som det gjør utvikling av brukergrensesnitt på toppen av databasen mer effektivt.

2.3.2 Beskrivelse av utviklingsmiljø

På grunn av valget om å bruke APEX må også Oracle database velges, siden APEX ikke kan installeres på noen annen database[5]. I en slik Oracle Database opprettes det gjerne et eller flere «workspace» med egendefinerte tillatelser, tillganger og databaser; nærmere bestemt databaseskjemaer. Disse skjemaene knytter eierforhold mellom databasebrukere og databaseobjekter, da for eksempel tabeller.

PL/SQL

Oracle Database er en relasjonsdatabase som benytter PL/SQL, som introduserer blant annet løkker, funksjoner og variabler til SQL. Altså kan databasespråket brukes til å løse problemstillinger som vanligvis ville kreve en tredjepart, i form av et annet programmeringspråk. Ved å velge Oracle Database får man på et vis alt i ett, ved at man bare har en enkelt tjener med et enkelt system som løser alt fra databaser til grensesnitt.

Oppdragsgiver har tilbydd å stille en database med APEX til disposisjon, og det vil her benyttes siste versjon (på dette tidspunktet) av både Oracle Database og APEX, henholdsvis versjon 12.2.0.1 og 5.1.4. Databasen er satt opp på en en virtuell maskin i skyen, som kjører en Linux distro, Ubuntu 16.04.04 LTS, på grunn av problemer rundt Server Name Identification (SNI) på Windows versjonen av Oracle Database.

[1] Som standard etter versjon 11.2.0.1 (11g)

Videre var det viktig for teamet å holde seg til færrest mulige aktører da det kom til valg av ytterlige verktøy, og det ble derfor forsøkt å benytte flest mulig produkt levert av Oracle i samsvar med Oracle-databasen og APEX. Som utviklingsverktøy ble SQL Developer valgt, da dette er et Oracle verktøy for utvikling av Oracle-databaser med PL/SQL[6]. Det finnes også integrasjon mot APEX-platformen i dette verktøyet.

Versjonskontroll

For versjonskontroll, og mulighet for samtidig arbeid i filer, ble GIT og repository på Bitbucket[1] valgt, dette fordi utviklerne er vant med verktøyene fra tidligere prosjekter i studiet. Etter utviklernes egen subjektive mening, hadde SQL Developer for dårlig integrasjon mot GIT, og hadde mangelfull eller for dårlig «code highlighting», i tillegg til at den ikke tillot editering på samme PL/SQL-script fra flere brukere i sanntid. Det ble derfor valgt å bruke teksteditoren Atom[2] som et ekstra skriveverktøy. Dette ble da brukt for å forenkle arbeidsprosessen i forhold til utvikling av kildekode. Forøvrig var det nødvendig å benytte SQL Developer for å få kompilert og «debugget» koden, og det endte dermed opp med å bli en del «klipp og lim» mellom de ulike verktøyene, for å holde begge oppdatert. Dette skapte naturlig nok noen komplikasjoner, og utviklerne har tatt selvkritikk angående dette.

2.3.3 Ressursbehov

Oppdragsgiver bistår utviklerteamet med å sette om tjeneren for løsningen. Videre må utviklerne ha mulighet til å holde en dialog med Aruba TAC. Dette kan teamet få tilgang til gjennom oppdragsgiver sin serviceavtale med Aruba. Utover dette vil utviklerteamet ta kontakt med aktuelle ressurspersoner ved behov. Dette kan for eksempel være faglærere ved NTNU i Gjøvik, andre medstudenter eller ressurser som teamet ellers kjenner.

[1] https://bitbucket.org/product
[2] https://atom.io

2.3.4 Testing

Teamet skal undervegs i utviklingen passe på å validere og verifisere systemet. Det skal jevnlig være en del av «Daily Scrum» å stille seg spørsmålene; «Bygges det riktige systemet?» og «Bygges systemet riktig?». Det vil også være viktig å utføre tester på inndataene som mottas fra API-et. Teamet skal også verifisere og validere koden for hver gang det legges til kode i Bitbucket. På den måten vil Bitbucket repositoriet alltid inneholde siste versjon av fungerende kode. Det vil være et felles ansvar at dette utføres, og hver enkelt skal sjekke sin egen og andres kode.

Det vil undervegs uføres «blackbox»-testing, både av utviklerteamet og av oppdragsgiver. Sistnevnte vil spesielt ha fokus på høynivå-testing, for eksempel legge til en ny kunde, for så å sjekke at den legges til i hovedoversikten. Teamet skal også undervegs utføre denne typen testing på inndata fra Aruba, for så å sjekke innholdet i tabellene.

Utviklerteamet vil undervegs utføre «whitebox»-testing. Dette vil foregå på både høynivå og lavnivå undervegs i utviklingen. Høynivå vil være viktig for å sørge for at funksjonene utfører den jobben de skal, men lavnivå må ikke overses. Selv om det finnes ferdige pakker for ulike oppgaver i Oracle, skal utviklerteamet selv forstå hvordan disse utfører oppgaven for å kvalitetssikre at rett fremgangsmåte er valgt.

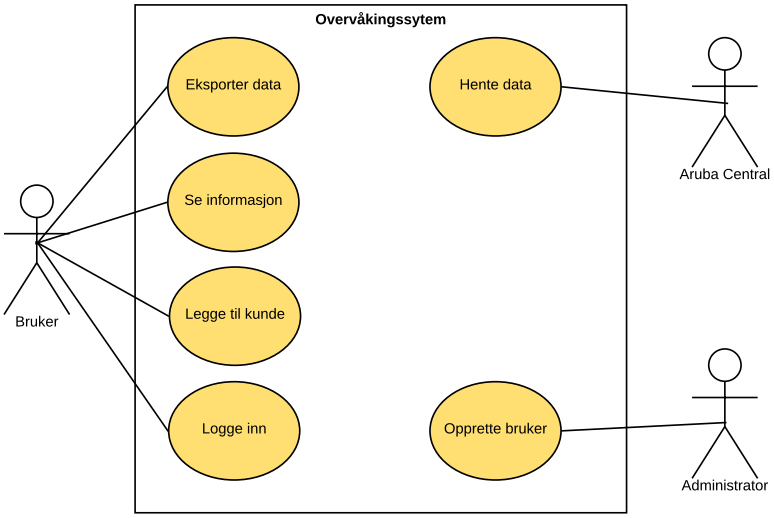
3 Kravspesifikasjon

I dette kapittelet vil det bli beskrevet nærmere i detalj hvordan løsningen skal fungere, i tillegg til ulike krav som skal sørge for at kvaliteten opprettholdes.

3.1 Funksjonelle krav

For å beskrive de handlinger som er ønsket i systemet, ble det laget flere use case basert på krav fra oppdragsgiver. Modellen vist i figur 3.1 viser da hele systemet, og aktørene som er involvert i hvert enkelt use case. Underseksjon 3.1.1 gir et noe bedre innsyn i aktørene, mens underseksjon 3.1.2 gir en grundig beskrivelse av alle use case.

3.1.1 Use Case modell



Figur 3.1: Use case modell

Aktører

- **Bruker**: En person i overvåkningsteam som administrer nettverkene.
- **Aruba Central**: Det eksterne API-et som systemet henter all informasjon om de forskjellige kundene fra.
- **Administrator**: Administratorbruker for systemet som legger inn brukertilganger.

3.1.2 Use Case

På grunn av prosjektets relativt få use case, beskrives alle på lavnivå. Noen av disse vil være forenklede da noe av funksjonaliteten er behandlet av verktøy i APEX-platformen, som er en del av systemet.

Use case	**Logge inn**
Aktør	Bruker
Hensikt	Autentisere en bruker i systemet.
Beskrivelse	For at bruker skal ha tilgang til systemet og de data dette skal vise til den aktuelle bruker, er brukeren nødt å logge seg inn med gyldig akkreditiver til innloggingen. Selve innloggingen gjøres gjennom APEX-platformens innloggingsløsning.
Forutsetninger	• Brukeren oppgir gyldige akkreditiver.
Ettervirkninger	Ingen spesifikke.

Hovedforløp	• Bruker oppgir korrekt akkreditiv i de angitte feltene. • Akkreditivene blir sjekket av APEX-løsningen. • Bruker blir korrekt autentisert i systemet.
Alternativt løp	• Bruker skriver inn manglende akkreditiv (for eksempel et tegn i passordet er feil). Se feilsituasjon 1. • Systemet sier ifra om manglene i GUI.
Feilsituasjoner	1. Bruker skriver inn ugyldig akkreditiver; Systemet sier ifra om feil i GUI-et og bruker må be administrator om nye akkreditiv.

Tabell 3.1: Logg inn use case

Use case	**Opprette bruker**
Aktør	Administrator
Hensikt	Administrator skal kunne opprette nye brukere i systemet.
Forutsetninger	Ingen spesielle.
Ettervirkninger	Ingen spesielle.

Hovedforløp	• Administrator logger inn i APEX. • Administrator oppretter en ny bruker til systemet. • Administrator gir en gitt person nødvendige informasjon om de nye privilegiene.
Alternativ løp	Ingen spesielle.
Feilsituasjoner	1. Administrator får ikke logget inn; Feilsøk systemet. 2. Administrator får ikke opprettet ny bruker; Systemet gir eventuelle feilmeldinger.

Tabell 3.2: Oprette ny bruker use case

Use case	**Se informasjon**
Aktør	Bruker
Hensikt	Presentere aktuelle data til bruker på en bestemt måte, slik at dette er «informasjon» og ikke «rådata».
Beskrivelse	Overvåkningsteamet skal kunne se relevant informasjon om kunder på en enkel og oversiktlig måte. Når noe går galt i et av kundenes nettverk skal teamet kunne få en overordnet forståelse av situasjonen ved å se på oversikten.
Forutsetninger	• Bruker må være innlogget.

Ettervirkninger	Påvirker ikke systemet.
Hovedforløp	• Bruker får epost om feil fra Aruba Centrals varslingssystem. • Bruker ser på kundeoversikten og finner mer detaljert info om feilen, og kan handle ut i fra dette.
Alternative løp	• Bruker får epost om feil fra Aruba Central varslingssystem. • Bruker ser på kundeoversikten, men finner ingen åpenbar grunn for varselen. • Bruker kan da sjekke systemloggen gjennom webapplikasjonen og handle basert på denne. • Om det ikke finnes feilmeldinger i loggen, må bruker vente i underkant av oppdateringsfrekvensen til systemet.
Feilsituasjoner	1. Systemet får ikke kontakt med databasen og viser ikke informasjon; Feilsøk systemet og/eller internett tilkobling.

Tabell 3.3: Se informasjon use case

Use case	**Legge til kunde**
Aktør	Bruker
Hensikt	Legge til ny kunde i databasen.

Beskrivelse	Bruker legger selv til hvilke kunder som skal ligge i databasen, og dermed vise i kundeoversikten. Det legges også inn parametere for hva som er viktig å følge med på for denne kunden. Dette tallet blir så brukt videre til å kalkulere et tall som kan sorteres på oversikten. På denne måten vil kunder med flest eller mest kritiske feil komme øverst i oversikten.
Forutsetninger	• Bruker må være innlogget.
Ettervirkninger	Informasjon om kunden blir hentet ved use case «Hent data», vist i tabell 3.5.
Hovedforløp	• Bruker velger ønsket handling i webapplikasjonen. • Bruker skriver inn nødvendig informasjon i alle påkrevde felter. • Systemet legger inn informasjonen i databasen og setter i gang use case «Hente data». • Bruker blir automatisk sendt tilbake til kundeoversikten, hvor kunden som er lagt til vil vise.
Alternativt løp	• Bruker velger ønsket handling i webapplikasjonen. • Bruker mangler et påkrevd felt, se feilsituasjon 1. • Bruker fyller inn feil informasjon i feltene. • Bruker blir automatisk sendt tilbake til kundeoversikten, hvor kunden som er lagt til ikke vil vise, se feilsituasjon 2.

Feilsituasjoner	1. Bruker mangler et påkrevd felt; Systemet gir feilmelding i GUI, bruker må rette opp feil. 2. Bruker fyller inn feil informasjon; Systemet vil ikke få hente data om den gitte kunden før dette er rettet med korrekte data. Bruker må sjekke systemlogg, og deretter editere de data som er gitt.

Tabell 3.4: Legge til kunde use case

Use case	**Hente data**
Aktør	Aruba Central
Hensikt	Hente data om kunder og deres nettverk fra Aruba Central gjennom API-et.
Beskrivelse	For at systemet skal holde seg oppdatert til en hver tid må det kontinuerlig hente data fra Aruba Central. Derfor henter systemet ønsket informasjon fra API-et ved hjelp av flere forskjellige forespørsler og legger disse i databasen.
Forutsetninger	• Aruba Centrals API må være tilgjengelig.
Ettervirkninger	Data hentet fra API-et vil ligge i databasen, slik at andre funksjoner i systemet kan bruke disse.

Hovedforløp	• Systemet gjør en forespørsel mot API-et. • API-et vurderer forespørselen, og svarer med ønsket data. • Systemet tar imot svaret og sjekker om det fikk ønsket data. • Systemet legger data inn i databasen.
Alternativt løp	• Systemet gjør en forespørsel mot API-et. • API-et vurderer forespørselen, men oppdager feil i denne og sender en feilmelding. • Systemet finner feilmeldingen. Se feilsituasjon 1. • Systemet tar i mot et gyldig svar og men finner ikke ønsket data. Se feilsituasjon 2. • Systemet legger ikke inn data i databasen.
Feilsituasjoner	1. API-et oppdager feil i en forespørsel og sender en feilmelding; Systemet logger meldingen, bruker må sjekke systemlogg. 2. Systemet finner ikke ønsket data i svaret fra API-et; Systemet logger feilen, bruker må sjekke systemlogg.

Tabell 3.5: Hente data use case

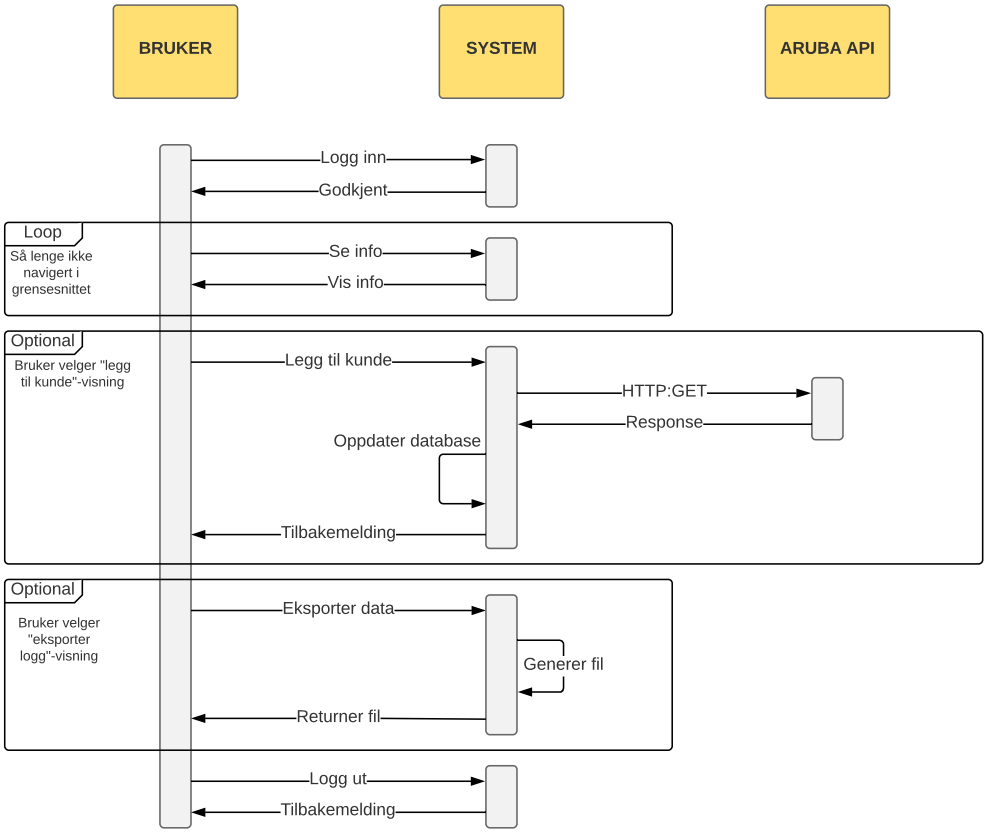
Use case	**Eksporter data**
Aktør	Bruker
Hensikt	Hente ut data i ønskede filformater.

Beskrivelse	I enkelte tilfeller ønsker bruker å hente ut rene lister over for eksempel alle enheter en kunde har i sine nettverk. Bruker kan da velge hvilken informasjon de vil se, og eksportere dette til forhåndsvalgte filformater.
Forutsetninger	• Databasen har data om kunder og deres nettverk. • Bruker må være logget inn.
Ettervirkninger	Ingen spesielle.
Hovedforløp	• Bruker velger ønsket funksjon i webapplikasjonen. • Bruker velger hvilke data de ønsker å eksportere. • Bruker velger filformat, og trykker send. • Systemet gir bruker en fil med ønsket data i.
Alternativt løp	• Bruker velger ønsket funksjon. • Bruker velger hvilke data de ønsker å eksportere. • Bruker trykker eksporter og velger filformat. • Systemet gir bruker en fil uten data. Se feilsituasjon 1.
Feilsituasjoner	1. Systemet gir bruker fil uten data; Feilsøk systemet

Tabell 3.6: Eksporter data use case

3.2 System sekvensdiagram

I stedet for flere system sekvensdiagram som illustrer hvert enkelt use case, viser figur 3.2 tiltenkt oppførsel til systemet under normal drift, med de funksjonaliteter en bruker har tilgjengelig. Det er også viktig å påpeke at systemet kontinuerlig skal oppdatere innhold i databasen tiltenkt brukergrensesnittet, uavhengig av interaksjon fra bruker. Dette illustreres i detalj i figur 4.3.



Figur 3.2: System sekvensdiagram for systemet i normal drift

3.3 Supplementære krav

Under vil videre krav til sytemet listes opp. Disse kravene er en sammensetning av initielle krav fra oppdragsgiver, og krav utviklerteamet har utledet undervegs etter videre samtaler med oppdragsgiver.

3.3.1 Brukergrensesnitt

- Ikoner og tekst i brukergrensesnittet skal være så store og «fyldige» at bruker kan sitte og overvåke en storskjerm(omtrent 30 tommer), fra en avstand på tre til fire meter uten å måtte anstrenge øynene.
- Brukergrensesnittet i hovedvinduet skal inneholde en tabell over kunder, hvor kunder med feil i nettverk vises øverst i tabellen.
- Kun kunder som oppdragsgiver velger selv, skal vises i grensesnittet.
- Navigering i brukergrensesnittet skal holdes til det absolutte minimum.
- Kundevisningen skal være sortert etter alvorlighetsgrad på nettverksfeil.
- Det bør eksistere egne faner for visning av nettverk og enheter, slik at det blir enkelt å eksportere informasjon om disse separat.

3.3.2 Operasjonelt

- Systemet under normal drift skal ikke trenge annet vedlikehold enn innlegging av nye Aruba Central-kunder.
- Informasjonen som vises skal kontinuerlig holdes oppdatert i databasen
- Systemet bør fungere som normalt, selv etter feil på nettverk, strømforsyning eller andre hendelser som medfører at webapplikajsonen mister kontakt med server.
- Feil i webapplikasjonen skal ikke føre til uopprettelige feil ellers i systemet, da hovedsakelig databasen.

- Det må eksistere en logg for feilrapportering, slik at driftpersonell kan enkelt feilsøke eventuelle problemer.
- Etter at autentiseringsprosessen mot systemet er gjennomført, skal brukere umiddelbart få tilgang til kundeoversikten.
- Systemet må takle skalering av antall kunder og nettverk/enheter under disse.

3.3.3 Sikkerhetsmessig

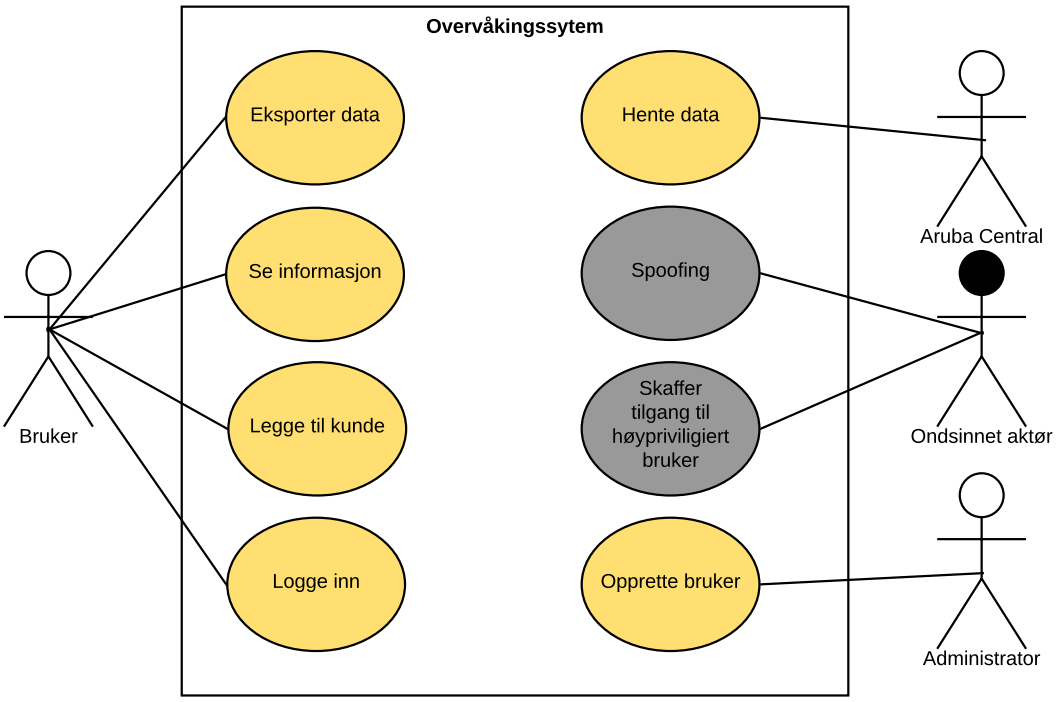
- En bruker skal ikke kunne se informasjon uten å autentisere seg.
- Alle variabler og konstanter[1] brukt mot API-et skal krypteres.
- Alle passord som lagres i databasen skal krypteres.
- Skal tilfredsstille krav satt av GDPR.

3.3.4 Dokumentasjon

- Det skal skrives dokumentasjon av systemet for bruk og enkel feilsøking.
- All kode skal dokumenteres for enkel videreutvikling.
- All kildekode skal legges tilgjengelig for oppdragsgiver i et GIT-repository.

[1]For eksempel: Tokens; Access token og Refresh token, Akkreditiv; client id og client secret

3.4 Misuse case



Figur 3.3: Misuse case for systemet

Misuse case	**Spoofing**
Aktør	Ondsinnet aktør
Hensikt	Få tak i personopplysninger ved å stjele akkreditiv og lignende fra systemets kommunikasjon.
Beskrivelse	Systemet spør API-et med hjelp av diverse akkreditiver. Aktøren avskjærer dataoverføringen og kopierer data fra denne.
Data	Akkreditiver for spørringer mot API-et.

Angrep	Den ondsinnede aktøren setter opp en spoof-server mellom systemets server og API-ets tjener, slik at systemet blir ledet til å tro at det kommuniserer med relevante mottakere

Tabell 3.7: Spoofing misuse case

Misuse case	**Skaffer tilgang til høypriviligert bruker**
Aktør	Ondsinnet aktør
Hensikt	Ødelegge for oppdragsgiver, eller for egen gagn
Beskrivelse	Den ondsinnete aktøren skaffer seg tilgang til en bruker med mange privilegier (for eksempel administratorbruker), og utnytter disse privilegiene til handlinger som gagner han/hun, eller ødelegger for oppdragsgiver.
Data	All data i databasen.
Angrep	Ondsinnet aktør skaffer seg tilgang til en høypriviligert bruker med tilgang til systemet. Angriperen bruker privilegiene til å slette eller manipulere databasen slik at systemet ikke lenger er funksjonelt.

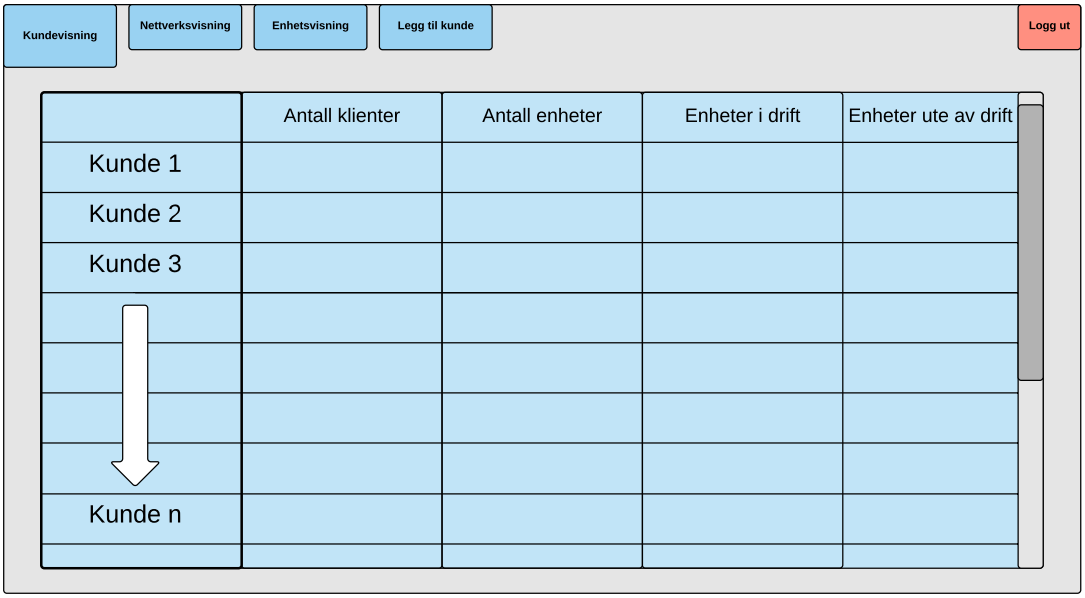
Tabell 3.8: Skaffer tilgang til høypriviligert bruker misuse case

4 Design

4.1 Grafisk

Hensikten med webapplikasjonen er at den skal være ergonomisk og enkel i bruk. Ergo skal det rent grafisk ikke være mye som endebruker må forholde seg til. Målet til oppdragsgiver er å få en god oversikt over de ulike kundene de administrerer nettverk for, slik at det blir enkelt å prioritere hvem og hvor det skal utføres tjenester og vedlikehold. I det øyeblikket en bruker logger inn, vil det første som dukker opp være en kundeoversikt. Derfra skal man kunne navigere til andre oversikter for å vise annen informasjon, eller foreta handlinger som for eksempel å legge til en kunde.

Figur 4.1 gir et oversiktsbilde over hvordan det tenkte brukergrensesnittet skal se ut for kundeoversikten.



Figur 4.1: Skisse av brukergrensesnitt

Kundevisningen skal sorteres etter noen bakomliggende parametre som defineres i prosessen som oppretter en ny kunde i databasen. Disse parameterne skal ha en verdi som tilsvarer alvorlighetsgrad på for eksempel enheter ute av drift, eller for høy bruk av båndbredde på et nettverk.

4.2 Arkitektur

Ved valg av arkitekturmodell må det vurderes flere viktige elementer. For eksempel skal systemet håndtere skalerbarhet med tanke på datamengde i databasen, og all transport av informasjon skal bli gjort på en sikker måte. Med utgangspunkt i disse kriteriene kunne «Pipe and Filter» vært en aktuell modell, da systemet er styrt av prosesser.

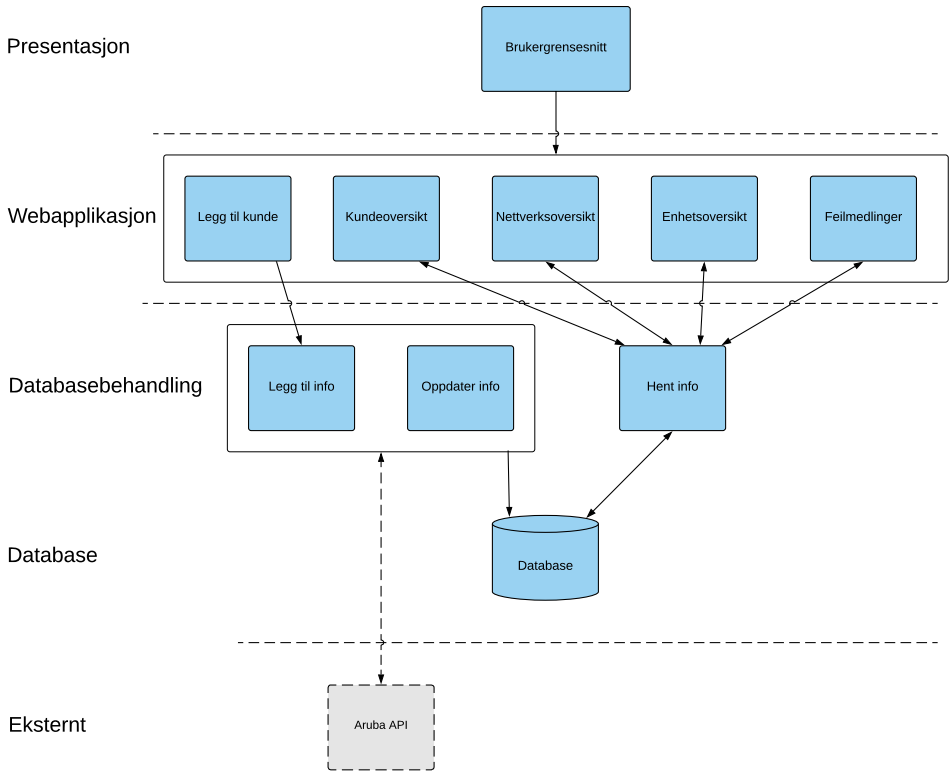
Det skal ved et gitt tidsintervall utføres en «oppdater informasjon»-prosedyre,

hvor konsekvensen blir at ny informasjonen vises på skjermen. Forespørsler etter informasjon kan nærmest ses på som en transaksjon. Bruker skal ikke styre denne prosessen i noe særlig grad, ei heller er prosessen styrt av handlinger/input undervegs, og derfor vil ikke denne modellen kunne dekke systemets behov.

Lagdelingsmodell

Systemet skal kunne hente informasjon fra databasen, og vil derfor trenge en tynn klient, og en tykk tjener. Videre vil informasjonen bli sendt gjennom flere lag før den kommer frem til den rette instansen, og derfor vil lagdelingsmodellen bli vurdert. En slik modell representerer systemet ved å dele det inn i forskjellige lag, der hvert lag tilbyr tjenester. Modellen baserer seg på at en base ligger i bunn, og at et grensesnitt ligger på toppen. Dette passer godt overens med modulene i systemet. Det er normalt at lagdelingsmodellen har et forretningslag, men i mindre systemer og i tilfeller når databasebehandlingslaget bruker for eksempel SQL, kan dette bygges inn i databasebehandlingslaget[7]. Selv om modellen ikke er god på skalering, begrenses dette på grunn av at skaleringen kun skjer på en isolert del av modellen. Derfor vil modellen håndtere skalering i god nok grad for dette systemet.

På grunn av et fokus som setter databasen høyt, benyttes et enkelt designmønster som illustrerer arkitekturen på best mulig måte. Lagdelingsmodellen vil gjøre det mulig å ivareta kravet om sikkerhet i prosessen og i lagene. På grunnlag av dette beskrives arkitekturen gjennom en lagdelingsmodell, som vist i figur 4.2. Denne modellen danner en oversikt over kommunikasjonsflyten mellom de ulike modulene i systemet. Alle de ulike visningene i brukergrensesnittet skal hente informasjon fra databasen, der informasjonen kontinuerlig holdes oppdatert.



Figur 4.2: Lagdelingsmodell

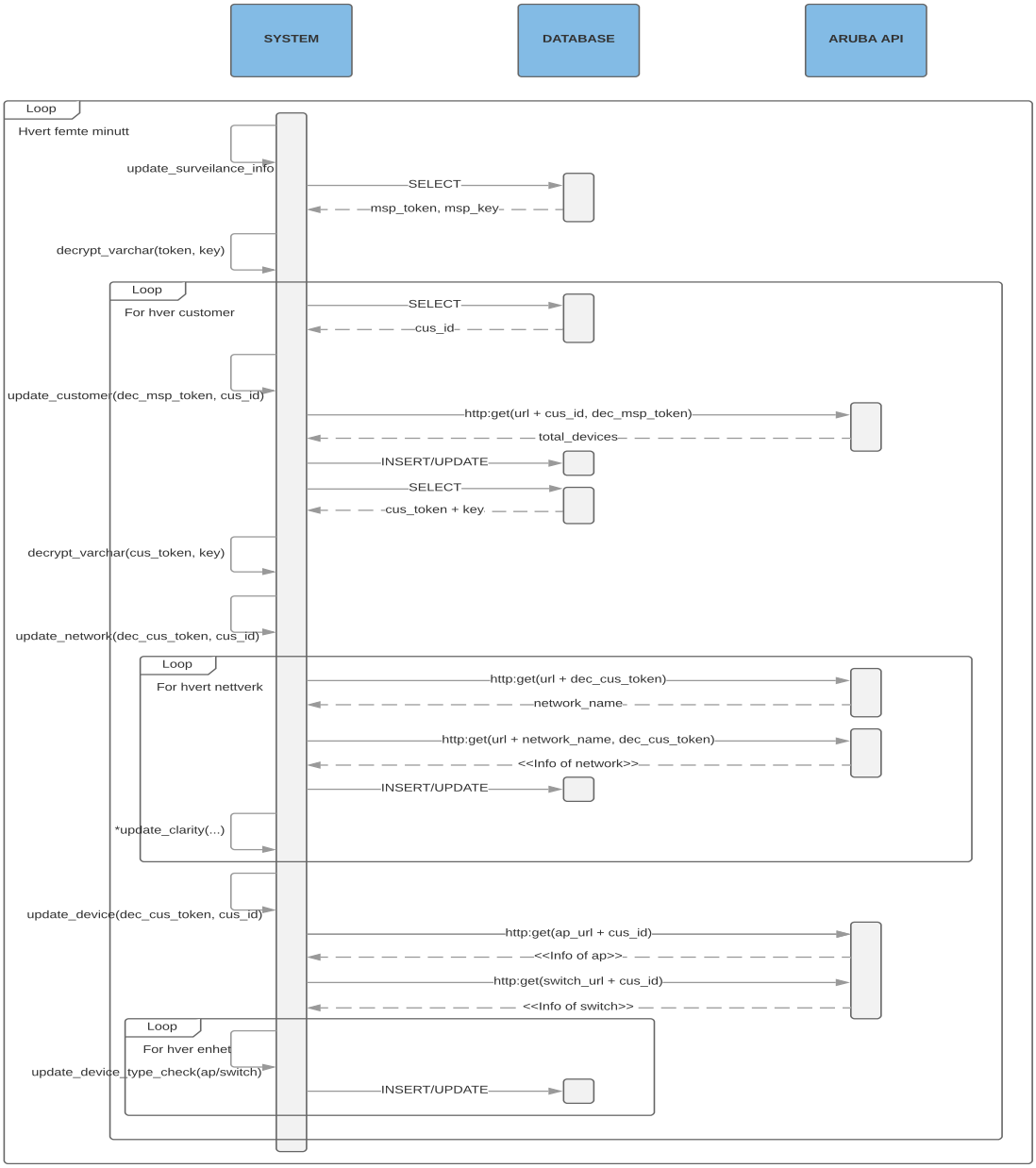
Den eksterne aktøren «Aruba API» er inkludert som en modul i lagdelingsmodellen, da denne er essensiell for å forstå hvordan informasjon skal hentes og lagres i databasen. Videre er det vist at databasebehandlingslaget oppdaterer info uavhengig av webapplikasjonen, og at alle forespørsler om informasjon henter dette via det samme laget.

4.3 Objektorientert design

Objektorientert design prøver å fremstille sammenhengen mellom objekter, gjerne ved hjelp av UML. Gjerne også hvordan objekter skal fungere og ikke hva objektene er, som ved use case.

4.3.1 Sekvensdiagram

Kompleksiteten rundt oppdatering av innhold i databasen gjorde det nødvendig å konstruere et sekvensdiagram for akkurat denne modulen av systemet. I figur 4.3 er en fullstendig representasjon av alle stegene fra systemet selv initierer oppdatering, til informasjonen om kunder, deres nettverk og tilhørende enheter er fullstendig oppdatert i databasen.



Figur 4.3: Sekvensdiagram for oppdatering av data

4.3.2 Designmønster

Ved valg av designmønster er det viktig å se på hva slags funksjonalitet som skal tilbys. Overvåkingssystemet skal hente informasjon gjennom det eksterne API-et, behandle disse og lagre det i systemets database. Informasjonen i databasen skal hovedsakelig vises i et oversiktsbilde for brukeren. Webapplikasjonen kan sees som «observatør» og informasjonen i databasen som «subjekt».

Observer pattern

Løsningen skal designes etter «Observer pattern», et designmønster beskrevet av Gang of Four[8]. Dette er et atferdsmessig designmønster som identifiserer felles kommunikasjonsmønstre mellom objekter, og realiserer disse. På denne måten vil kommunikasjonen mellom observatør og subjekt være definert. Observer pattern har en ulempe med at strukturen kan bli noe komplisert, men fordelene er mange.

GRASP

Observer pattern har også flere overførbare sammenhenger mot ulike prinsipper innenfor General Responsibility Assignment Software Patterns (GRASP)[9]. Observatør er «Creator», og har ansvaret for å lage nye instanser av «Subject». Dette skal gjøres av bruker i GUI og kan for eksempel være å legge til en ny kunde i en tabell. Observatørrollen skal ha kontroll over informasjonen, men Subject har også noe ansvar, i form av å holde seg oppdatert med en oppdateringsfunsksjon.

Videre i samme eksempel, kan det hos Subject eksistere respektive triggere for hver tabell som blir kjørt eventuelt ved behov, og «rollback» hvis en funksjon/prosedyre ikke blir fullstendig gjennomført. Databasen vil derfor være «Information Expert» i systemet, men hvis mønsteret blir brukt riktig, skal også observatør ha riktig informasjon. Det er APEX som vil inneha «Controller»-rollen, fordi den både mottar og koordinerer de ulike kontrollene som sytemet opererer på. APEX vil representere det overordnede systemet og jobber direkte ovenfor Oracle-databaser, og vil derfor også defineres som en «fasadekontroller».

5 Implementasjon og Realisering

Arbeidsmetodikken har i stor grad basert seg på prinsipper fra Scrum(seksjon 2.1) og teamet erfarte at dette var et meget godt valg. Grunnet få spesifikke krav fra oppdragsgiver initielt, og heller en beskrivelse av behov, ble tilpasningsdyktighet sentralt for arbeidsmetodikken. Det ble tidlig etterspurt om oppdragsgiver hadde forslag til plattform som løsningen kunne utvikles i, og det ble da presentert andre plattformer enn hva som initielt var planlagt. Grunnet god og konsekvent kontakt med oppdragsgiver, fikk teamet tilpasset den nye løsningen etter behov som ble avdekket undervegs i utviklingen.

For å muliggjøre disse ønskene, ble fokuset satt til databasen og de prosedyrer og funksjoner som må lages for å holde databasen oppdatert og skalerbar. Hvordan dette er blitt utført står nærmere forklart i seksjon 5.4.4. På denne måten ble det sikret at ekstra tiltenkt funksjonalitet kan implementeres forholdsvis enkelt ved videreutvikling.

5.1 Oracle Application Express

APEX har et enkelt brukergrensesnitt og god hjelpefunksjonalitet. For eksempel ved oppretting av en ny side for webapplikasjonen følger brukeren et trinnbasert oppsett hvor man trykker seg frem til ønsket funksjonalitet. Opprettelsen er som følger: Først velges hvilken type side det skal være, ut i fra ett gitt antall maler, for eksempel rapport, kalender, med flere. Videre velges funksjonalitet ut i fra menyer som kommer undervegs i prosessen. Om man ønsker editering av den gitte malen, finnes det også flere ferdige moduler som kan brukes på siden. Løsningen som er utviklet baserer seg på «interactive grid» fra malen «report», og henter data fra databasen ved hjelp av SQL spørringer. SQL spørringen i hovedoversikten «Overview» henter informasjon fra de fleste av tabellene i databasen, aggregerer på flere av dataene, og er som vist i kodelisting 1.

```
select AWS_CUSTOMER_DETAIL.AWS_CUD_CUSTOMER_ID as ID,
       AWS_CUSTOMER_DETAIL.AWS_CUD_NAME as Name,
       count(AWS_NETWORK.AWS_NET_CUS_ID) as Networks,
       sum(AWS_NETWORK.AWS_NET_CLIENT_COUNT) as Total_Clients,
       AWS_CUSTOMER.AWS_CUS_TOTAL_DEVICES as Registered_Devices,
       check_down_devices(AWS_CUD_CUSTOMER_ID, 'ap', 'Up') AS Up_ap,
       check_down_devices(AWS_CUD_CUSTOMER_ID, 'ap', 'Down') AS Down_ap,
       check_down_devices(AWS_CUD_CUSTOMER_ID, 'switch', 'Up') AS Up_switch,
       check_down_devices(AWS_CUD_CUSTOMER_ID, 'switch', 'Down') AS Down_switch,
       calculate_risk(AWS_CUSTOMER_DETAIL.AWS_CUD_CUSTOMER_ID) AS Risk,
       AVG(AWS_NETWORK.AWS_NET_RECIEVED) as Recieved,
       AVG(AWS_NETWORK.AWS_NET_TRANSMITTED) as Transmitted,
       AVG(AWS_CLARITY.AWS_CLA_DHCP_SCORE) as DHCP_score,
       AVG(AWS_CLARITY.AWS_CLA_DNS_SCORE)  as DNS_score,
       AVG(AWS_CLARITY.AWS_CLA_AUTH_SCORE) as AUTH_score
 from  AWS_NETWORK,
       AWS_CUSTOMER,
       AWS_CUSTOMER_DETAIL,
       AWS_CLARITY
 where AWS_CUSTOMER_DETAIL.AWS_CUD_ID = AWS_CUSTOMER.AWS_CUS_CUD_ID
   and AWS_NETWORK.AWS_NET_CUS_ID     = AWS_CUSTOMER.AWS_CUS_ID
   and AWS_CLARITY.AWS_CLA_NETWORK_ID = AWS_NETWORK.AWS_NET_ID
 group by AWS_CUSTOMER_DETAIL.AWS_CUD_CUSTOMER_ID,
          AWS_CUSTOMER_DETAIL.AWS_CUD_NAME,
          AWS_CUSTOMER.AWS_CUS_TOTAL_DEVICES
```

Kodelisting 1: SQL-spørring som henter informasjon til kundevisning(hovedvisning/overview), kjøres hvert 5 min

På lignende vis blir også lister over nettverk, enheter og logg hentet ut. Alle sider i webapplikasjonen kan bare hente data, unntatt «Edit Customer»-siden. I «Edit Customer» vises alle kundene som er lagt inn i systemet, i en tabell. På siden kan man endre og legge til nye kunder via et tilknyttet skjema. Dette er enkelt satt opp i APEX.

5.2 Aruba Central REST-API

En stor del av dette prosjektet har gått ut på å bruke Aruba Centrals REST-API for å hente ut data for lagring i databasen. I denne seksjonen blir det forklart hvordan generell kommunikasjon mot dette API-et foregår og litt

bakgrunnsinformasjon som danner et grunnlag for ord og uttrykk.

5.2.1 Autentisering mot Aruba Centrals API

For å benytte API-et til Aruba Central må man autentisere seg ved hjelp av OAuth2 og Aruba Centrals «TLS-endpoint». Gjennom dette endpoint-et kan systemet autentisere seg som en gitt Aruba Central bruker og få en autentiseringskode som kan byttes inn til en access token. Ved hjelp av et slikt token kan overvåkingssystemet utføre spørringer på vegne av den nevnte Aruba Central brukeren.

Systemet benytter den gitte Aruba Central brukerens akkreditiv for autentisering mot API-et. Om akkreditivene er gyldige, gir API-et systemet informasjonskapsler som validerer koblingen mellom systemet og Aruba Central gjennom HTTPS. Videre tar systemet i bruk informasjonskapslene og en «client id»-kode tilknyttet en gitt «aruba central kunde»[1] til å hente autentiseringskode.

Når systemet har fått en autentiseringskode, må det skaffes en access token for den gitte Aruba Central kunden, for å gjøre spørringer mot API-et. For å veksle inn autentiseringskoden til access token må systemet også bruke en kode ved navn «client secret» som tilhører den spesifikke kunden. Dette sikrer at en kunde ikke kan spørre om informasjon tilhørende andre kunder i Aruba Central. Om alt går som det skal, vil systemet få både en access token og en refresh token som vil kunne benyttes videre. Denne prosessen skjer hovedsakelig ved systemets første oppstart, etter systemet har hatt en krasj eller når en gitt access token ikke er gyldig lengre.

5.2.2 Fornyelse av access token

Grunnet sikkerhetsmessige årsaker, har Aruba satt gyldigheten av access tokens og refresh tokens til to timer. Dette fører til at systemet må fornye disse innen dette tidsvinduet for å unngå å gjennomføre autentiseringsprosessen på nytt (som forklart i 5.2.1).

[1] Samlebegrep for alle kundekontoene til oppdragsgiver, inklusiv Managed Service Provider (MSP)-bruker

For å gjennomføre en fornyelse av access- og refresh-token trenger man bare sistnevnte, da denne er gyldig som autorisasjon på lik linje med access token i denne sammenhengen. Ved suksessfull gjennomføring av operasjonen, vil systemet få nye access- og refresh-token, som da brukes videre ved neste spørring mot API-et.

5.2.3 Utføre en spørring mot API-et

Systemet er som nevnt utviklet for en Oracle-database ved hjelp av script-språket PL/SQL. For å utføre spørringer fra serveren, som står bak oppdragsgivers brannmur, måtte det legges inn sertifikat brukt i sammenheng med API-et i databasens wallet. Her var det ikke nok med bare wildcard-sertifikatet[1] for endpoint-et. Etter mye testing med hjelp fra oppdragsgivers DBA ble det oppdaget at alle sertifikater brukt mot API-et måtte være i wallet for at systemet skulle fungere. Sertifikatene ble funnet og eksportert ved hjelp av nettleser, og videreført til databaseadministrator for å få lagt til sertifikatene til wallet.

Det oppsto flere problemer i sammenheng med kommunikasjonen med API-et. Årsaken var at Arubas server brukte SNI[2]. Versjonen av Oracle-databasen som systemet kjørte på da, hadde en «bug» som gjorde at SNI ikke fungerte. Dette ble løst ved å oppdatere databaseversjonen på systemets server. Hvordan teamet håndterte denne typen problematikk er beskrevet nærmere i kapittel om utførsel 7.

Da kommunikasjonen kom på plass, kunne funksjoner i pakken «UTL_HTTP» for PL/SQL brukes til spørringer over HTTPS mot API-et. Pakken definerer metoder for å utføre spørringer gjennom HTTPS og HTTP, men er nødt til å ha tilgang til sertifikatet i wallet for kommunikasjon over HTTPS. Systemet bruker bare GET og POST metodene under HTTP-protokollen i forespørslene mot API-et og alle svar fra API-et kommer i form av JSON. Et generisk eksempel på en slik spørring er som vist i kodelisting 2.

[1] Et sertifikat som gjelder for alle domener som slutter på et gitt suffiks, for eksempel *.domene.com

[2] Utvidelse av TLS protokollen hvor en klient indikerer hvilken «host» den prøver å koble til i starten av «handshake»

```
DECLARE
  req utl_http.req;
  resp utl_http.resp;

BEGIN
  -- Makes the wallet available
  -- This way one can issue requests to https enpoints
  utl_http.set_wallet(wallet_path, wallet_pasword);

  req := utl_http.begin_request(
    https://app1-apigw.central.arubanetworks.com/<some endpoint>,
    'GET',
    'HTTP/1,1'
  );

  -- retrieve response
  resp := utl_http.get_response(req);
END
```

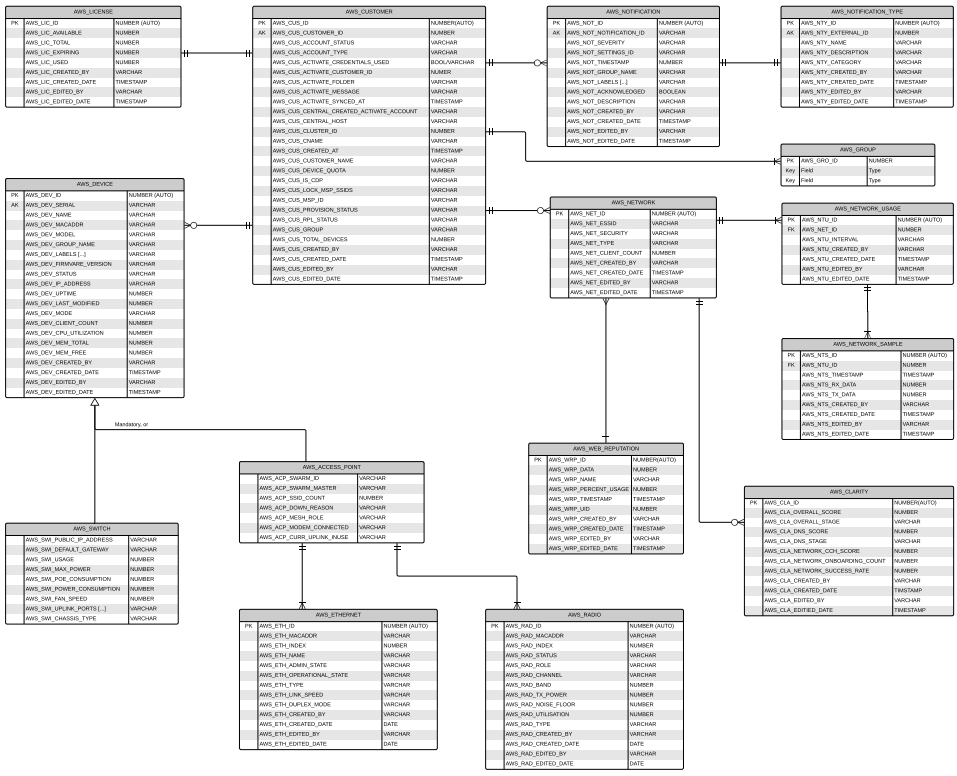
Kodelisting 2: Eksempel på GET forespørsel mot API-et

5.3 Database

Oppdragsgiver hadde opprinnelig et ønske om å ha ekstra funksjonalitet rundt løsningen utover det å bare kunne vise informasjon som hentes fra API-et. Derfor ble det vurdert som nødvendig å modellerere og implementere en database for systemet. På denne måten ble det mulig å mellomlagre data slik at man kan for eksempel lagre historikk, vise statistikk i ulike diagrammer og eksportere data til ønskede filformater.

I starten ble det laget en databasemodell basert på hvilke data som var mulig å hente ut fra Aruba Central via API-et, og i tillegg som en måte å kartlegge disse dataene på for teamets egen del. Figur 5.1 viser hvordan den opprinnelige modellen så ut.

Etter utarbeidelse av denne modellen, hadde utviklerteamet og oppdragsgiver



Figur 5.1: Initiell modellering av database

et mer utdypende kravspesifikasjonsmøte på hvordan kundeoversikten skulle se ut og hva den skulle inneholde. Kravet ble da en tabell med en kunde per rad og hvor kolonnene skulle inneholde:

Kunde ID Arubas id på en gitt kunde

Navn Navnet på kunden

Antall nettverk Antall nettverk kunden har

Antall klienter Antall tilkoblede klienter på alle nettverkene til kunden

DHCP-resultat Poengsum på hvor god tilkoblingen og tildeling av ip-adresser er mot klientene

DNS-resultat Poengsum på hvor bra oppslagene er mot DNS-server

Auth-resultat Poengsum på hvor bra autentisering av klienter er

Registrerte enheter Alle enheter en kunde har registrert til seg

Ap oppe Antall aksesspunkter som er i drift

Ap nede Antall aksesspunkter som er ute av drift

Switch oppe Antall switcher som er i drift

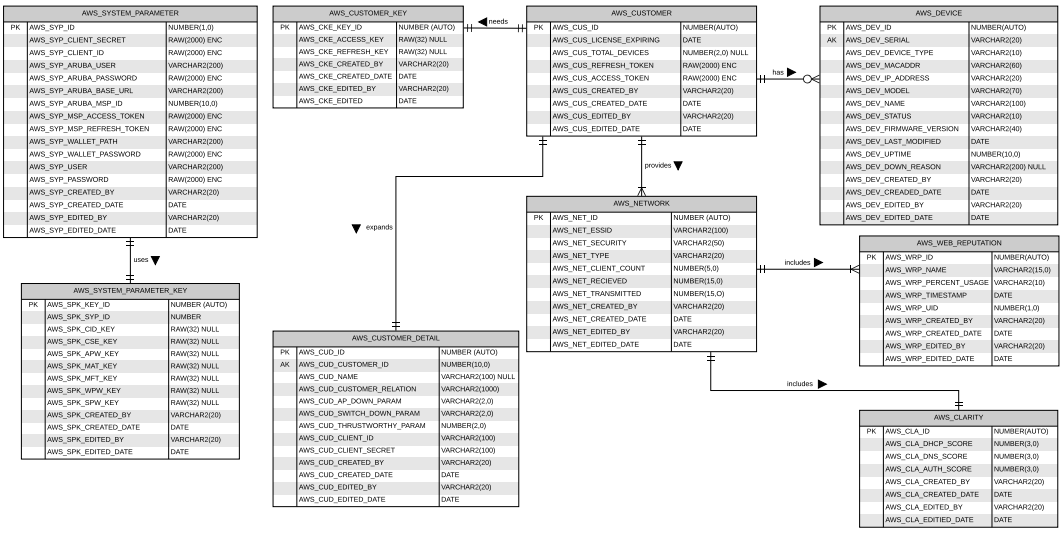
Switch nede Antall switcher som er ute av drift

Gj. motatt data Gjennomsnittlig Mb/s nedlastet siste døgn, som sum av alle nettverk til kunden

Gj. sendt data Gjennomsnittlig Mb/s opplastet siste døgn, som sum av alle nettverk til kunden

Databasen måtte altså holde data slik at disse punktene kunne oppfylles. Dette gjorde at modellen gjennomgikk en kraftig reduksjon og revisjon, som resulterte i modellen vist i figur 5.2. NETWORK_USAGE og NETWORK_SAMPLE tabellene ble fjernet og det ble istedet lagt til to attributter i NETWORK for mengde data opplastet og nedlastet. DEVICE sin arv ble skrinlagt og bare tabellen selv ble beholdt, men med noen mindre endringer av attributtene. For å skille mellom en switch og et aksesspunkt ble attributtet DEVICE_TYPE brukt, som inneholder stringen «switch» eller «ap». Førsteutkastet hadde også med NOTIFICATION og GROUP, men disse ble overflødige og dermed fjernet etter spesifiseringene fra oppdragsgiver.

Prosessen førte også til at tabeller for krypteringsnøkler ble inkludert i modellen. Enkelte attributter i CUSTOMER og SYSTEM_PARAMETER er krypterte og har derfor tilhørende tabeller som lagrer nøklene brukt for kryptering. Tabellen for systemparametere (SYSTEM_PARAMETER) holder på konstanter brukt av systemet ved flere anledninger, for eksempel «path» til wallet. En annen fordel med dette, er hvis noen av disse parameterene må endres, slipper man å endre dette mer enn en enkelt plass for at koden skal fortsette å fungere.



Figur 5.2: Endelig modell over databasen

5.3.1 Normalisering

Databasen er normalisert slik at duplikate verdier ikke skal forekomme. For å sørge for at alle tabellene oppfylte første normalform, var det viktig å passe på at det kun eksisterer atomære[1] verdier. Dette ble løst ved å passe på hvert enkelt attributt i en tabell har en unik verdi. Databasen oppfyller også krav om andre normalform, ved at alle felter som ikke er nøkler er avhengig av nøkkelfeltene(PK). Tredje normalform er oppfylt ved at ingen attributter som ikke er nøkler, ikke er avhengig av andre felter som ikke er nøkler. For å tilfredstille disse kravene ble for det hver kunde laget en egen tabell, både for nettverk, og for enheter.

5.3.2 Struktur

Under listes alle tabellene i databasen med en kort forklaring av deres formål.

SYSTEM_PARAMETER Holder akkreditiver for MSP, for eksempel «Path» og passord for wallet.

[1]Verdier som består av bare en enkelt verdi

SYSTEM_PARAMETER_KEY Krypteringsnøkler for de krypterte verdiene i SYSTEM_PARAMETER.

CUSTOMER Holder all data som bruker ikke må legge inn om en kunde, blant annet krypterte tokens for den kunden. Denne tabellen danner kjernen for databasen, fordi all informasjon er knyttet til en gitt kunde, dens nettverk eller dens enheter.

CUSTOMER_KEY Krypteringsnøkler for de krypterte verdiene i CUSTOMER. Nøklene er unike for hver kunde.

CUSTOMER_DETAIL Holder all informasjon som må bli gitt av bruker om en gitt kunde. De tre _PARAM attributtene settes av bruker for å si hva som skal prioriteres som viktig for denne kunden.

NETWORK Inneholder navn på nettverk, antall tilkoblede klienter og mengde nettverkstrafikk på nettverket i løpet av foregående døgn. Alle nettverk tilhører en gitt kunde.

WEB_REPUTATION Holder på kategorisert nettverkstrafikk med 6 nivåer som sier noe om påliteligheten til domene som aksesseres i et nettverk.

DEVICE Holder all informasjon om en enhet under en gitt kunde.

CLARITY Holder informasjon om et gitt nettverks helse(DHCP-, DNS- og autentiserings-resultat).

LOG Holder alle loggmeldinger overvåkingssystemet generer.

Det var ønskelig å unngå «null-verdier» i databasen så langt dette lot seg gjøre, og som konsekvens av dette ble CUSTOMER delt i to. Den ene for data som bruker legger inn manuelt(CUSTOMER_DETAIL), mens den andre holder data som systemet henter automatisk(CUSTOMER). I enkelte tilfeller ble det derimot ikke hensiktsmessig å forby «null-verdier» i alle attributter, da for eksempel DOWN_STATUS-attributtet i DEVICE er «null» når den gitte enheten er i normal drift.

Alle tabellene i databasen unntatt SYSTEM_PARAMETER-tabellene og LOG har som nevnt en form for relasjon til CUSTOMER. En kunde har flere nettverk og flere enheter. Hvert nettverk har i tillegg informasjon om sin egen «helse»(CLARITY) og sin type nettversktrafikk(WEB_REPUTATION).

5.4 Implementasjon i henhold til krav

5.4.1 Funksjonalitet

Denne seksjonen tar utgangspunkt i definerte use case fra kapittel 3, Kravspesifikasjon.

Logg inn

Oppdragsgiver hadde som krav at det skal eksistere en side en bruker kan logge seg inn i systemet fra. Dette ble enkelt gjennomført ved å bruke APEX sin innebygde mal «logg inn-side», som inneholder to tekstfelt hvor brukeren kan skrive inn gyldig brukernavn og passord(5.4.5), og en knapp. Knappens funksjonalitet er å gi en bruker tilgang til systemet, gitt at gyldige akkreditiv er oppgitt. APEX kan autentisere brukere på forskjellige måter, for dette systemet ble det valgt å bruke «Application Express Accounts». Brukerene av løsningen må være lagt inn med en konto i tilhørende APEX-workspace.

En kort beskrivelse av hvordan legge til ny bruker og hvilke rettigheter denne trenger til webapplikasjonen er lagt inn i brukermanualen(vedlegg F). Det er bare administrator i APEX som kan legge til nye brukere.

Se informasjon

Utviklerteamet ville bruke tekst og ikoner da dette ble vurdert som den beste måten å fremstille informasjon på, og at disse kan skaleres og tilpasses i forhold til skjermstørrelse. På bakgrunn av dette var det bedre å implementere et grafisk brukergrensesnitt i motsetning til for eksempel et kommandolinjegrensesnitt. Her kom APEX-verktøyet for brukergrensesnitt godt til nytte. Verktøyet kan generere et enkelt og oversiktlig grensesnitt med «dra og slipp», i motsetning til å måtte formatere alt i HTML og CSS. Mer om overvåkingssystemets grensesnitt kommer i seksjon 5.4.3.

Legge til kunde

Oppdragsgiver drifter nødvendigvis ikke nettverkene til alle kundene under NaaS-avtale, og det ble da nødvendig å implementere funksjonalitet slik at oppdragsgiver selv kan bestemme hvilke kunder som skal vises i kundeoversikten. Dette ble løst med en egen side med et typisk skjema i webapplikasjonen der brukeren kan legge inn ønsket kunde. I skjemaet må brukeren legge inn nødvendig kundeinformasjon for spørringer mot API-et, og dermed kan systemet hente informasjon om denne kunden.

Hente data

For å hente data benyttes Aruba Central, hvor informasjon om alle kunder under NaaS-avtalen er oppbevart. Dataene kan da hentes med Aruba Centrals REST-API. Store deler av utviklingen gikk med på å utvikle funksjoner som fylte disse tabellene med data fra API-et, i tillegg til funksjonalitet beskrevet rundt selve API-et(5.2).

For hver tabell i databasen er det laget en funksjon som henter data fra API-et, sjekker databasen for duplikater, og oppretter eller endrer rader i den gitte tabellen. DEVICE-tabellen er brukt for å gi et detaljert eksempel på hvordan slik funksjonalitet er oppnådd gjennom en funksjon. En gjennomgang av koden for dette eksempelet kommer i seksjon 5.4.2, og er ment å gi leser en forståelse av hvordan hver enkelt funksjon for tabellene er bygget opp.

Eksporter data

Innebygd funksjonalitet i APEX er benyttet for å lage et alternativ for eksportering av data. Det eksisterer mulighet for å eksportere alle tabellvisninger i webapplikasjonen til aktuelt filformat, hvor det for dette systemet ble valgt mulighet for CSV etter ønske fra oppdragsgiver. I tillegg finnes mulighet for eksportering av HTML.

Opprette bruker

Administrator av APEX-workspace kan opprette bruker(e) med begrensede privilegier som kan brukes til å logge inn i overvåkingssystemet og se de forskjellige tabellene. Opprettelse av en ny bruker gjøres i admininstrator-innstillingene i APEX.

5.4.2 Gjennomgang av eksempelkode

Denne seksjonen går detaljert gjennom et eksempel på en funksjon som henter data fra API-et, sjekker databasen for duplikater og handler ut i fra dette.

GET forespørsel

Alle GET-forespørsler til API-et blir gjort som vist i den eksplisitte funksjonen for utførelse av GET-forespørsler (kodelisting 3). Ved hjelp av denne får man innholdet i «body», de faktiske data til responsen, som et stort tekstobjekt, klar for tolking til «JSON-object»[1]. Her tilbyr APEX en pakke for PL/SQL med navn «APEX_JSON» som enkelt tar tekster formatert som JSON og oppretter JSON-objekt ut fra teksten. På denne måten kan det hentes data lettere enn å lete seg gjennom det originale tekstobjektet.

[1] I realiteten er dette en assosiativ liste(array), men vil bli omtalt som «objektet» for enkelhetens skyld

```
1  CREATE OR REPLACE FUNCTION get_https_response (
2      p_url IN VARCHAR2       -- any full API-url
3    ) IS
4
5    req       utl_http.req;  -- keeps track of the request
6    resp      utl_http.resp; -- keeps track of the response
7    temptext  CLOB;          -- used to read response
8    respvalue CLOB;          -- contains the full response
9
10   -- Wallet is needed for https-requests
11   utl_http.set_wallet(wallet_path, wallet_password);
12
13   -- builds the request, and gets the response
14   req := utl_http.begin_request(p_url, 'GET', 'HTTP/1.1');
15   resp := utl_http.get_response(req);
16
17   BEGIN
18     LOOP
19       -- loop through the whole response, line for line
20       -- and append each line to 'respvalue'
21       utl_http.read_line(r => resp, data => temptext);
22       respvalue := respvalue || temptext;
23     END LOOP;
24
25     EXEPTION
26     WHEN utl_http.end_of_body THEN  -- stop when the response
27       utl_http.end_response(resp);  -- has ended
28   END;
29
30   RETURN respvalue;
31 END get_https_response;
```

Kodelisting 3: Funksjon for GET-forespørsler, noe forenklet

Funksjonsheader

En funksjon som oppdaterer DEVICE-tabellen i databasen er relativt stor, og vil derfor stykkes opp i flere mindre biter. Først defineres selve funksjonen og dens parametere (4).

```
1  CREATE OR REPLACE PROCEDURE update_device (
2      p_cus_id IN NUMBER,      -- Aruba customer id
3      p_type VARCHAR2          -- type of device (ap/switch)
4    ) IS
```

Kodelisting 4: Funksjonsheader for update_device

Videre blir forespørsler mot API-et kjørt, før svarene blir tolket til JSON-objekter. Forespørslene mot API-et vil i dette tilfellet returnere en liste over alle enheter av den gitte typen som en kunde disponerer. Antallet enheter varierer stort mellom de forskjellige kundene til oppdragsgiver.

Tolking til JSON

Siden man nå har en liste av JSON-objekter som representerer enheter, kan man iterere gjennom alle disse, og hente ut informasjon fra hvert enkelt JSON-objekt, som vist i kodelisting 5.

```
1  -- the path in the JSON is slightly different depending on which 'type'
2  IF p_type = 'ap' THEN
3    v_req_type = 'aps';
4  END;
5
6  FOR v_device in apex_json.get_members(
7                        p_path => v_req_type,
8                        p_values => v_jsonvalue
9                      );
10 LOOP
11   -- retreive data for each field in DB
12   v_serialNum :=  apex_json.get_varchar2(
13                     p_values => v_device,
14                     p_path => 'serial'
15                   );
16
17   --  *
18   --  *
19
20   v_status    :=  apex_json.get_varchar2(
21                     p_values => v_device,
22                     p_path => 'status'
23                   );
24  );
```

Kodelisting 5: Parsing av JSON for update_device

Når dette er gjort for samtlige felter i JSON-objektet som sammenfaller med et felt i databasen, kan man begynne å opprette nye og/eller oppdatere eksisterende rader. For at samme funksjon skal kunne gjøre begge deler, trenger den spesiell funksjonalitet. Her har Oracle-databasen en funksjonalitet som kalles «MERGE». Denne er tiltenkt å sammenligne forskjellige tabeller og oppdatere eller opprette nye rader basert på et uttrykk utvikleren bestemmer. Selv om det her ikke sammenlignes to faktiske tabeller, kan man formulere uttrykket slik at enheter som ikke finnes i tabellen fra før, blir lagt til som ny rad, og at motsatte tilfeller blir oppdaterte.

Merge i løkke

Samtidig vil man sikre at data faktisk blir lagt inn i tabellene, selv om flere prosesser prøver å utføre handlinger på samme tabell. Noen ganger vil en av prosessene feile og dermed ikke utføre sine instrukser til det fulle. Dette kan løses ved å bruke en «uendelig løkke», som bare stopper når all data er lagt inn. Teamet hentet idéen om merge i en løkke fra stackoverflow [1].

En kombinasjon av MERGE og løkke, utvidet med operasjoner mot databasen, resulterer i en kodesnutt som vist i kodelisting 6.

Det er også verdt å nevne unntaksbehandlingen som er foretatt på slutten av kodelistingen. I Oracle-databaser vil det bli «kastet» unntak når operasjoner feiler, for eksempel ved sletting av en rad som ikke eksisterer i en gitt tabell. Unntak som «kastes» av databasen fanges opp av unntaksbehandleren, som er indikert med nøkkelordet «EXCEPTION» i kodesnutten. For å behandle et enkelt unntak, angir man dette med en eller flere «WHEN»-utsagn. MERGE-løkken fanger her opp både når gitt data ikke er funnet, og når gitt data er duplikat av eksisterende rader i tabellen. I denne situasjonen betyr disse unntakene henholdsvis samtidig sletting og samtidig innsetting.

Tabellen kan se forskjellig ut når MERGE-sjekken skjer og når faktisk innsetting eller oppdatering skjer, derfor kan det oppstå situasjoner hvor samtidig sletting og/eller samtidig innsetting skjer i mellomtiden. Da vil en av de nevnte unntakene forekomme og bli fanget opp. Unntaksbehandleren vil ikke gjøre noe med det, da man ønsker at loopen skal gå inntill data er satt inn eller oppdatert.

[1] link til stackoverflow: https://stackoverflow.com/questions/237327/oracle-how-to-upsert-update-or-insert-into-a-table/22777749?sfb=2#22777749

```
BEGIN
    LOOP
      BEGIN
        MERGE INTO AWS_DEVICE USING dual
        ON ("AWS_DEV_SERIAL" = v_serialNum)

        WHEN MATCHED THEN UPDATE SET (
          "AWS_DEV_DEVICE_TYPE" = p_type, *, *,
          "AWS_DEV_STATUS" = v_status
        )

        WHEN NOT MATCHED THEN INSERT (
          "AWS_DEV_CUS_ID", "AWS_DEV_SERIAL",
          *, *, "AWS_DEV_STATUS"
        )
        VALUES (
          v_table_cus_id, v_serialNum,
          *, *, v_status
        );

        EXIT; -- success? -> exit loop

    -- handle exceptions
    EXCEPTION
      -- the entry was concurrently deleted
      WHEN NO_DATA_FOUND THEN
        -- exception? -> no op, i.e. continue looping
        NULL;

      -- an entry was concurrently inserted
      WHEN DUP_VAL_ON_INDEX THEN
        -- exception? -> no op, i.e. continue looping
        NULL;
    END;
  END LOOP;
END;
```

Kodelisting 6: Løkke som utfører MERGE, og legger data inn i databasen

En sammensatt/fullstendig versjon av «update_device»-funksjonen vises i vedlegg E.1, om det er ønskelig å studere koden nærmere.

5.4.3 Grafisk brukergrensesnitt

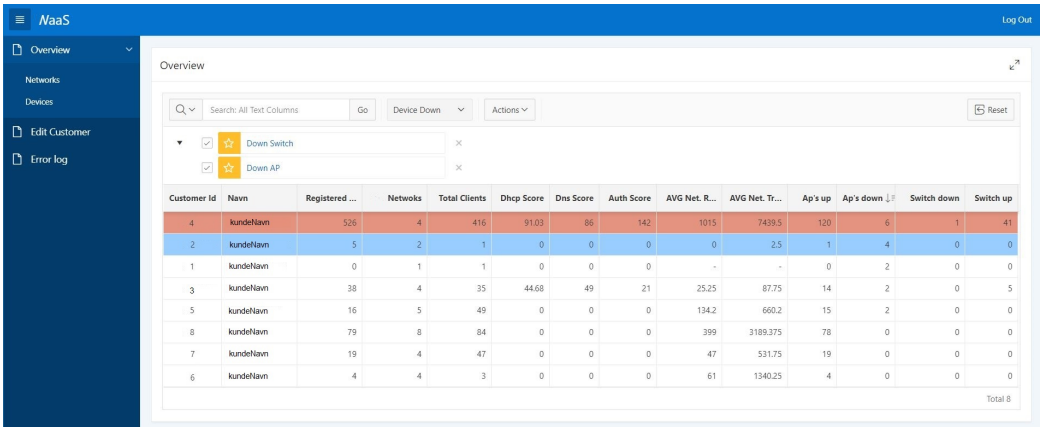
Siden det var ønskelig å konstruere et grensesnitt som skal vise mest mulig informasjon om en kundes nettverk i et og samme vindu, ble det en utfordring å sørge for god leselighet for overvåkningsteam som sitter tre til fire meter unna skjermen. Det var tidlig diskutert om det skulle implementeres grafiske fremstillinger i form av grafer og diagram, men denne idéen ble raskt nedstemt da det ville tatt for mye plass i forhold til annen relevant informasjon, og det er lite skalerbart på en enkelt skjerm. Systemet endte dermed opp med å inneholde en oversiktlig tabell i et passende stort format som viser oversikt over alle kundene og antall enheter som inngår, med tilhørende status(i/ute av drift). Kravet om at kun kunder som har forespurt å inngå i overvåkingssystemet dekkes gjennom «legg til kunde»-funksjonaliteten(3.4).

Kalkulere prioritet

For å løse kravet om en prioritert kundeliste bruker systemet en algoritme som regner ut en sum, som kundeoversikten kan sorteres på. Algoritmen benytter seg av parametere bruker har spesifisert ved oppretting av ny kunde. Ved å kalkulere prioriteten på de ulike kundene, vil kravet om en prioritert kundeliste dekkes. Navigering i grensesnittet er ikke nødvendig, med mindre en bruker ønsker å legge til flere kunder i systemet, eller få en oversikt over samtlige enheter eller nettverk.

APEX skalerer sine grensesnitt etter hvilken skjerm de blir presentert på, og tilfredstiller derfor kravet om at informasjonen i fremstillingen er lesbar fra tre til fire meter

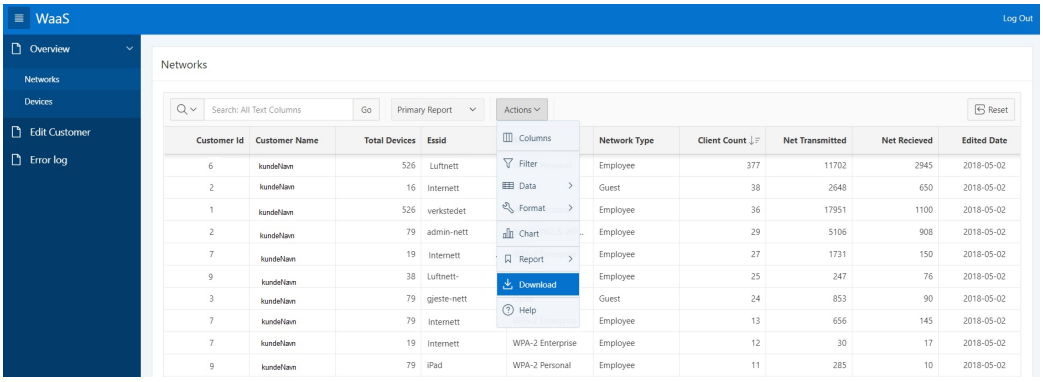
Grensesnittet 5.3 har en enkel meny på venstresiden som viser hvilke oversikter som kan vises. Menypunktene «Overview», «Networks», «Devices» og «Error Log», har henholdsvis hver sin oversikt som er fremstilt som en tabell av typen «Interactive Grid»[10]. Denne type tabell har et kraftfullt verktøy som tillater sluttbrukeren å sortere, definere og editere tabellene etter behov. Som vist i skjermbildet 5.3 kan sluttbrukeren for eksempel fargemarkere de



Figur 5.3: Skjermbilde av Løsningen - Oversiktsbilde

kundene som er av spesiell interesse i tabellen. I bildet er kunder med switcher ute av drift markert med rødt, og en kunde med flere enn fire aksesspunkt ute av drift markert med blått.

Disse egendefinerte tabellene kan lagres per bruker, eller for alle brukere av systemet, dersom dette er ønskelig. Bruker kan laste ned tabellen, selv etter manipulering av denne gjennom beskrevne handlinger. Funksjonaliteten er som vist i 5.4, og alternativene for filtype er CSV og HTML. Ytterligere ønsker fra oppdragsgiver resulterte i to andre views: «Network» og «Devices». Da disse var på plass, kunne oppdragsgiver enkelt eksportere nyttig data om nettverk og eneheter for enkelte kunder av spesiell interesse.



Figur 5.4: Skjermbilde for en nedlasting av nettverksliste

5.4.4 Operasjonelt

Informasjonen som vises i grensesnittet skal kontinuerlig holdes oppdatert. Dette vil si at informasjonen som ligger i databasen må være tilsvarende. Dette er løst ved at en oppdateringsfunksjon[1] som er koblet mot en scheduled job kjøres hvert femte minutt. Denne relativt lave oppdateringsfrekvensen ble valgt for å begrense antall spørringer mot API-et for ikke å overbruke serveren til Aruba Central, og at nettverkstrafikken ikke skulle bli for stor. Det var heller ikke nødvendig for oppdragsgiver å hente informasjon med en høyere frekvens. For at spørringene skulle fungere, var det også essensielt at hver kunde i databasen har en gyldig access token, slik at de forble autentisert[2]. Ved en eventuelt systemkrasj, og hvis jobben som kontinuerlig oppdaterer alle access tokens feiler, vil det bli kjørt ny autentisering for alle kunder.

5.4.5 Sikkerhet

Det var tydelig blant alle aktørene i prosjektet at sikkerhet skulle stå i fokus, med tanke på EU-loven GDPR som vil tre i kraft kort tid etter prosjektets slutt. Selv om utviklerne ikke hadde spesiell erfaring innen sikkerhet, var de opptatt av dette, og ønsket å gjøre vurderinger etter beste evne for å sikre systemet.

Sikkerhet i APEX

APEX har flere sikkerhetstiltak innebygget i platformen. Her er standard innstillinger benyttet, unntatt det som kalles «idle time». Sikkerhetstiltakene omfatter blant annet at man ikke kan gjenoppta «sessions», forhindrer ondsinnede aktører å tukle med URL-er og at hurtigbuffring ikke er tillat. APEX forhindrer også «cross site scripting» ved å unngå «tolking» av HTML i for eksempel tekstfelter og ved at det forhindres manipulering av URL-er. Videre tilbys det gode valideringsløsninger for skjema, slik at angripere ikke kan bruke dette som innfallsvinkel.

[1] Denne foretar nye spørringer mot API-et og oppdaterer innholdet i databasen(Hente data under 5.4.1)

[2] access tokens holdes oppdatert gjennom en egen funksjon som kjøres hver time, slik som forklart i seksjon 5.2.2

Adgang til systemet

Tilgang til databasen er forskjellig fra tillgang til webapplikasjonen, og er begrenset til en enkelt administratorbruker, som oppdragsgiver disponerer. Brukere opprettet i APEX for webapplikasjonen har i utgangspunktet ikke en databasebruker tilknyttet seg. Derimot har APEX-platformen selv en generell databasebruker som gjør spørringer på vegne av APEX-brukere. Platformens bruker har navnet «APEX_PUBLIC_USER» og er en del av Oracle-databasens «PUBLIC»-gruppe.

PUBLIC-gruppen har visse roller og/eller privilegier tilknyttet seg. Alle brukere i en Oracle-database vil alltid være en del av PUBLIC-gruppen, og vil da også ha de roller og privilegier som gruppen har, tilknyttet brukeren selv. Det er anbefalt at denne gruppen ikke har mer enn det absolutte minimum av tillatelser, som oftest innebærer SELECT-utsagn og EXECUTE-privilegier[1], som er normalverdier[11]. På denne måten sikres det at ikke hvem som helst kan endre databasens innhold fra webapplikasjonen.

Eksempelsvis i forhold til kryptering, må lagring av nøkler være sikkert. Nøklene er lagret i tabeller i samme database som resten av dataene i systemet, men er ikke tilgjengelig for brukere av webapplikasjonen på grunn av ovennevnte restriksjoner. Alle spørringer mot databasen som blir gjort av webapplikasjonen, må i tillegg være forhåndsdefinerte, og bruker av webapplikasjonen har ikke mulighet til å opprette nye/flere spørringer.

Altså kan alle brukere opprettet i APEX utføre forhåndsbestemte SQL-spørringer laget av utviklerteamet. Det er da ikke mulig for noen APEX-bruker å aksessere tabellene som lagrer nøkler, på grunn av at SQL-spørringer mot disse tabellene ikke eksisterer.

Kryptering

Et av kravene definert for overvåkingssystemet, var at alle variabler og konstanter knyttet til kommunikasjon mot Aruba Centrals API skulle krypteres. Disse variablene er, for å nevne noen: «client id», «client secret» og «access token». Videre skulle passord som lagres i databasen krypteres.

[1] Lar bruker eksekvere funksjoner

Til dette tilbyr Oracle-databaser en pakke kalt «DBMS_CRYPTO». Pakken har flere kjente, og mye brukte krypteringsskjema og metoder for kryptering[12]. Ved hjelp av pakken kunne det lages egne metoder for kryptering og dekryptering. Kodelisting 7 viser hvordan dekryptering er utført i overvåkingssystemet. Alle nøkler i systemet er 256 bit lang, altså 32 bytes, og er generert pseudo-random i systemet.

Pseudo-random nøkler for kryptering genereres slik

```
dbms_crypto.randombytes(v_bytes);
```

Der `v_bytes` er antall bytes som skal genereres. Disse skal i følge Oracle[13] være sikre, kryptografisk sett. For ytterligere sikkerhet, skiftes nøklene hver gang et felt i databasen blir byttet ut, da for eksempel ved et «UPDATE»-utsagn. Nøklene er også generert unike for hvert enkelt felt som skal krypteres. Dette gjør at nøklene som brukes ikke er gyldige «for alltid», og i det tilfellet en ondsinnet aktør får tak i en nøkkel, vil denne bare gi aktøren tilgang til det enkelte feltet. Enda en fordel med dette er at en ondsinnet aktør som oftest må få tilgang til tre unike nøkler for å gjøre forespørsler mot API-et ved hjelp av stjålet data fra systemet, grunnet API-ets autentiseringsprotokoll.

Som forklart i Adgang til systemet er det ikke mulig å aksessere tabellene som holder nøkler gjennom APEX, og teamet har da vurdert at nøklene er sikre.

```
1  CREATE OR REPLACE FUNCTION decrypt_varchar (
2      p_encrypted_string IN RAW,  -- an encrypted VARCHAR2, as RAW
3      p_decryption_key   IN RAW   -- the key used to encrypt the VARCHAR2
4    ) RETURN VARCHAR2 IS
5
6    v_decrypted_raw RAW(2000);
7    v_decrypted_string VARCHAR2(100);
8
9    -- Total encryption type. This has to be equal for
10   -- both the encryption and decryption functions
11   v_encryption_type  PLS_INTEGER :=
12                            DBMS_CRYPTO.ENCRYPT_AES256
13                          + DBMS_CRYPTO.CHAIN_CBC
14                          + DBMS_CRYPTO.PAD_PKCS5;
15
16 BEGIN
17   -- gets the raw result after decryption
18   v_decrypted_raw := dbms_crypto.decrypt(
19     src => p_encrypted_string,
20     typ => v_encryption_type,
21     key => p_decryption_key
22   );
23
24   -- "translates" RAW to VARCHAR2
25   v_decrypted_string := utl_i18n.raw_to_char(
26     data => p_decrypted_raw
27   );
28
29   -- return the now decrypted and "translated" VARCHAR2
30   RETURN v_decrypted_string;
31 END;
```

Kodelisting 7: Eksempel på en dekrypteringsfunksjon

GDPR

Den nye loven om personvern fra EU ble spesielt påpekt av oppdragsgiver som noe de ville overvåkingssystemet skulle ta høyde for. Dette på grunn av at enkelte spørringer mot API-et kan gi en respons med sensitiv informasjon om nettaktiviteten til enkeltbrukere på et gitt nettverk. Etter samtale med oppdragsgiver ble det utarbeidet hvilken informasjon de ønsket å vise i systemet, og teamet kunne dermed fastslå at de ønskede dataene ikke inneholder personopplysninger definert av Datatilsynet[14].

Sikkerhet i forhold til missuse case

Spoofing
For å forhindre spoofing på kommunikasjon, er det implementert tiltak mot spoofing via nettverk. Nettverksmessig skjer all trafikk mot API-et gjennom HTTPS, som vil si at all informasjon i spørringen er kryptert. Selv om en ondsinnet aktør klarer å spoofe nettverkstrafikken vil ikke denne være mulig å dekryptere på grunn av TLS-protokollens funksjonaliteter.[1]

Skaffer tilgang til høypriviligert bruker
På grunn av dataens art, er det ikke kritisk om databasen eventuelt blir slettet i sin helhet, så lenge kildekoden er tilgjengelig. Det finnes PL/SQL-script som oppretter alle tabeller og tilhørende triggere, fremmednøkler og funksjoner. I teorien kan databasen opprettes fra grunnen av i løpet av få minutter, slik at bruker kan opprette kundene på ny. All nødvendig data vil da bli hentet fortløpende, og systemet vil være funksjonelt etter denne prosessen. Dette er selvfølgelig aldri en god ting, men vil ikke ha de aller største negative konsekvensene.

Om en ondsinnet aktør derimot prøver å få tilgang til akkreditiv for autentisering mot Aruba Centrals API kan dette ha større konsekvenser. Ved et slikt tilfelle vil en aktør kunne gi seg ut for å være systemet, og dermed utføre spørringer på vegne av dette. Her vil angriper kunne få tak i enkelte persondata om brukere på kundenes nettverk hos oppdragsgiver.

For at disse scenariene skal forekomme, må administratorbrukeren som oppdragsgiver distribuerer bli «tatt over» av en ondsinnet aktør. Dette kan fore-

[1] Les mer om TLS her.

komme på flere måter, men blir i hovedsak tatt høyde for av oppdragsgiver ved god praksis med tanke på passordbehandling.

6 Testing og kvalitetssikring

6.1 Testing

På bakgrunn av prosjektets natur, hvor det stadig dukket opp endringer eller tilføyinger til kravspesifikasjon, falt det naturlig å benytte en egendefinert versjon av v-modellen. Måten dette fungerte på gjennom hele prosjektperioden, var at med en gang det ble innført nye krav, ville utviklerne validere produktet i sin helhet, for så å vurdere hvilke endringer dette ville medføre og på hvilken måtte det nye kravet kunne integreres med resten av systemet.

overvåkningsteamet har utført en akseptansetest av overvåkingssystemet. Det ble da gjennomført vanlige driftsrelaterte handlinger, som for eksempel å legge til en ny kunde, for å verifisere at systemet fungerte i normal bruk. Det ble utført enkelte stresstester på systemet ved å utføre spørringene ti ganger så hyppig som systemet er instilt på nå. Som teamet mistenkte ligger flaskehalsen rundt spørringene opp mot det eksterne API-et, da disse bruker lang tid (dvs ca 10- 20 sekunder) på å respondere. Hypotetisk sett kan prosesseringsevnen til serveren, eventuelt nettverket, stå til slutt som flaskehals, da SQL-databasen kan takle relativt store mengder med data.

6.2 Produktkvalitet

Kvaliteten til prosjektet er i stor grad ivaretatt ved at utviklingen er foretatt på en plattform som oppdragsgiver selv har foreslått, og med tilsvarende verktøy. Dette har gjort det enkelt å skreddersy løsningen med komponenter som de tiltenkte brukerne er vandt med, samt god oppfølgning på effektiv og ressurssparende implementasjon for utviklernes del. Kommunikasjon mellom lagdelene(se figur 4.2) fungerer også optimalt grunnet konsekvent bruk av Oracle-verktøy.

Videre er det blitt implementert logging av feil, slik at hvis noe går galt et sted under operasjon, vil det bli fanget opp et unntak og skrevet ut en feilmelding til bruker. Dette er ment for å gjøre feilsøking enklere. I tillegg er det utarbeidet en brukermanual som skal fungere som et hjelpemiddel for nye

brukere av systemet, i tillegg til å inneholde en enkel guide for de vanligeste feilmeldingene. Manualen ligger vedlagt i vedlegg F.

Kildekode

All kildekode er nøye kommentert med forklaring og hensikt til de respektive kodesnuttene. Dette sørger for at personell som ønsker å videreutvikle, og som nylig er blitt introdusert til systemet, enkelt kan sette seg inn i funksjonaliteten til hver eneste funksjon og prosedyre. Utviklerene har i størst mulig grad prøvd å følge «best practice» for at koden skal ha en allment godkjent struktur og språk, da med fokus på variabel- og funksjonsnavn.

APEX

APEX har flere funksjonaliteter som sikrer kvaliteten til produktet. Den har innebygde sjekker, validering og mer. For eksempel finnes det sjekker for at variabler ikke blir duplisert, den vil gi melding dersom SQL-spørringer ikke lengre er gyldige og om en side blir opprettet feilaktig eller ikke i tråd med andre innstillinger. APEX har til og med en egen «Advisor» som kan kjøres og vil blant annet sjekke om etter feil i kode, sikkerhet, oppsett og mer. Denne funksjonaliteten har utviklerteamet tatt i bruk og implementert i systemet.

Løsningen er også internasjonalisert, i utviklingsfasen ble løsningen laget på engelsk. APEX har funksjonalitet som skanner hele løsningen for ord/ textvariabler, konverter dette til en xlf og laster den ned. Teamet har oversatt denne fra engelsk til norsk, og lastet den opp igjen til APEX som så legger den oversatte versjonen på en kopi av løsningen. På den måten er løsningen internasjonalisert og fungerer i dag etter det språket som datamaskinen er satt til mellom norsk og engelsk. Engelsk er i utgangspunktet satt som standard.

6.3 Prosesskvalitet

For å sikre god flyt i prosjektet og utviklingsprosessen, samt å oppnå et godt resultat, ble det i planleggingsfasen konstruert et gantt-skjema(Figur

C.5), samt satt opp møte- og milepælsdatoer. Disse fungerte som faste holdepunkt for utviklerene gjennom prosjektet og det ble dermed sikret at samtlige medlemmer var underforstått med hvor man sto i prosessen. De forskjellige møtene som ble holdt underveis var også en faktor som hjalp i denne sammenheng.

Scrum-verktøyet Taiga, i tillegg til «daily scrums», var også med på å sikre prosessens kvalitet gjennom hver enkelt sprint. Her opererte teamet med et typisk «sprint taskboard» for hver sprint der oppgaver ble opprettet som «product backlog items» og gjerne tildelt en eller flere utviklere. Verktøyet hjalp teamet med å strukturere arbeidet og holde «flyten» i prosessen gående ved at utviklerene kunne velge oppgaver fritt fra de som var tilgjengelige, eventuelt gi assistanse på andres oppgaver.

6.3.1 Møtevirksomhet

Møtevirksomhet har vært et sentralt aspekt gjennom hele prosjektperioden. Gruppen har dratt stor nytte av å avholde mange møter internt, med veileder og med oppdragsgiver. Alle substansielle møter er blitt fortløpende ført referat for, slik at en fullstendig dokumentert og god arbeidsprosess er blitt ivaretatt. De ulike møtereferatene omtalt under, ligger vedlagt i H.

Daily scrum

Under hele prosjektperioden har det blitt holdt daily scrum-møter tilnærmet hver eneste dag det er blitt utført arbeid på systemet, men også i enkelte tilfeller hvor rapportskriving har stått i fokus. Referatene har vært tilgjengelige for hele teamet under prosjektet, slik at utviklerene kunne holde seg oppdatert ved fravær, samt for å kunne se tilbake på prosessen for å orientere seg om hvilke arbeidsoppgaver som var blitt ferdigstilt.

Retrospective

Det har vært viktig for utviklerne å avholde retrospective-møter etter hver sprint for å kartlegge arbeidsprosessen kontinuerlig i et mer overordnet perspektiv. Under disse møtene ble det diskutert og dokumentert både positive

og negative elementer som omhandlet den gitte sprinten, samt ført opp forslag til tiltak for å forbedre sistnevnte, slik at videre arbeidsprosess ble så effektivisert som mulig.

Veiledning

Gruppen har brukt veiledning aktivt under hele prosjektet for å sikre et kontinuerlig fokus på veien mot et ferdig produkt. Hovedtema under disse møtene har hovedsaklig vært rapportstruktur og innhold, men har også til tider fungert som teknisk veiledning av ulik karakter. Utenom dette er det blitt diskutert generelle problemstillinger og forslag til tiltak rundt uforutsette problemer og lignende.

Oppdragsgiver

Den kanskje viktigste kommunikasjonskanalen til utviklerne har vært ansatte hos oppdragsgiver som har bistått med rammeverk, tekniske tips og «best practice» innen bransjerelaterte utviklingsverktøy, som for eksempel APEX. Disse møtene hadde som oftest relativt lite formelt innhold og var for det meste dialoger om rent tekniske problemstillinger. Noen av møtene har derimot vært mer omfattende gjennomganger i form av kravspesifikasjonsmøter og innføring i utviklingsverktøy. Disse er det blitt ført referat av som referansepunkt til rapporten i vedlegg H.2

7 Utførelse

Dette kapittelet inneholder sammendrag fra hver enkel periode i prosjektet, og de erfaringer som gruppen har opparbeidet seg. Kapittelet beskriver prosessen i forhold til Scrum og respektive sprinter.

7.1 Oppstart

Periode: 08.01 - 26.01
Total tidsbruk: 159 timer

Bacheloroppgaven startet offisielt med underskriving av kontrakt med oppdragsgiver den 09.01.18. Uken etter fikk gruppen teknisk innføring i hvordan EVRY Norge AS tilbyr NaaS og hva de opprinnelig brukte som overvåkningsverktøy. Oppdragsgiver hadde tidligere prøvd å lage et system for å bedre overvåkningen, men ikke kommet noen vei med å få kontakt med API-et. Videre gikk tiden med på å prosjektere og skrive prosjektplan.

7.2 Scrum-prosessen

Periode: 29.01 - 27.04
Total tidsbruk: 1103 timer

7.2.1 Sprint 1

Periode: 29.01 - 09.02
Total tidsbruk: 198 timer

Hovedfokuset til den første sprinten i prosjektet var å få kontakt med API-et, og det ble brukt mye tid på å teste ulike forespørsler.

Allerede første dag i Sprint 1 hadde teamet et møte (referat: H.2.3) med en ansatt hos oppdragsgiver, hvor han forklarte hva som hadde blitt prøvd tidligere og hva problematikken med å få kontakt med Aruba sitt API var per

dags dato. Teamet ble oppfordret til å bruke verktøyet postman for å utforske oppbyggingen av API-et og for å teste ulike GET- og POST-spørringer.

Det første som ble oppdaget, var at Aruba Central faktisk har to forskjellige API som de kaller «internal»- og «production»-API. Etter å ha testet begge to, konkluderte utviklerne med at det var production API som var korrekt.

Tokens

Videre var det en utfordring angående hvordan bruke tokens mot Aruba Central. Etterhvert ble det funnet ut hvordan «client id» og «client secret» blir generert, for så å bruke disse i forespørsler etter tokens. Dette sørget for at kontakt med API-et ble etablert, allikevel var det fremdeles problemer med å motta ønsket respons. Teamet sendte e-post til Aruba TAC for å spørre om hvorfor de fleste av GET-forespørslene som ble prøvd ga en tom respons, selv om «response code» var «200 - OK». E-postkorrespondansen med TAC-team startet 29. januar, og e-post ble utvekslet hyppig utover sprint-perioden.

Feilsøking

Teamet hadde da grunn til å tro at det eksisterer ulike nivå for aksessering av informasjon gjennom Aruba Central API-et, og prøvde å beskrive dette i e-poster til TAC. I sammenheng med dette ble det i tillegg ført flere telefonsamtaler med ingeniører i Aruba TAC-team uten å finne en umiddelbar løsning på problemet.

Siden denne sprinten var sterkt preget av mye feilsøking og venting på svar, måtte utviklerne finne andre ting å gjøre i mellomtiden. I hovedsak gikk dette ut på å lære seg de nye rammeverkene som var tenkt skulle brukes i utviklingen. Blant annet ble det konstruert et førsteutkast til «side for innlogging» på Node.js rammeverket, noe som var en del av den opprinnelige planen.

7.2.2 Sprint 2

Periode: 12.02 - 23.02
Total tidsbruk: 149 timer

I Sprint 2 var fokuset fortsatt å oppnå «korrekt» kontakt med API-et, samtidig som det var viktig å flytte mer fokus over på utvikling av selve webapplikasjonen.

Feilsøking

Det gikk med mye tid på feilsøking angående hvorfor API-et ikke ga forventet, eller til tider ingen informasjon i det hele tatt etter forespørsler som teamet gikk ut i fra var korrekte. Utviklerne var forberedt på at dette ville ta tid, og fulgte da tiltak fra ROS-analysen (seksjon 2.2.2) og jobbet parallelt med å kartlegge hva slags informasjon som potensielt kunne hentes ut fra API-et. Aruba Central har brukt verktøyet Swagger[1] for å generere dokumentasjon til API-et sitt, med forhåndsdefinerte URL-er hvor brukeren selv kan legge inn resterende nødvendige variable som legges til disse. Gjennom dette verktøyet kunne teamet teste forespørselene og se hva slags respons disse ga. For eksempel baserte teamet seg til tider på responseksempel for de ulike forespørselene, men det viste seg å ikke alltid stemme overens med den faktiske responsen motatt fra API-et.

API Gateway

Etter e-post korrespondanse med Aruba TAC-team, samt noe testing og feiling, kunne utviklerne 20. februar konkludere med at for hver enkelt kunde det skal kjøres API-spørring mot, må ha API Gateway[2] aktivert. Teamet konferte da med oppdragsgiver og bestile aktivering av API Gateway omgående. Dette ble aktivert 01.03, da kunne endelig utvikling av funksjonalitet i databasen starte, ettersom denne var avhengig av fungerende kommunikasjon med API-et. Videre fantes det flere «seksjoner» i API-Gateway, deriblant «monitoring» som da inneholde spørringer om overvåkningsdata. I hver seksjon

[1] Et verktøy for utvikling av API. https://swagger.io/
[2] Del av en stor applikasjon, her Aruba Central, pakket inn som en frittstående applikasjon. Her finnes dokumentasjon for API-et, utviklet gjennom Swagger.

fantes API-spørringer innunder den spesifikke seksjonen, som gjerne omfattet et gitt tema.

Samtidig som denne problematikken pågikk, jobbet teamet med å sette opp utviklingsmiljøet, inkludert APEX. Her ble det brukt en del tid på å sette seg inn i APEX, hvordan hente informasjon fra databasen for så å vise denne i webapplikasjonen. Gjennom APEX møtte teamet på flere problemer i form av tilkobling mot API-et, blant annet sertifikatproblemer. Som nevnt i kapittel 5, må sertifikatene til Aruba Central bli lagt inn i wallet av oppdragsgiver, for så å kunne bli tatt i bruk via APEX. Da sertifikatet var på plass, ble et nytt problem oppdaget.

Kommunikasjon

Hypotesen denne gangen var at falt Aruba brukte Server Name Identification (SNI). Teamet ble informert av en ansatt hos oppdragsgiver at Oracle hadde en «bug» ved bruk av SNI i databaseversjonen, 11.2.0.1 (11g), som da var i bruk. For å løse dette var det nødvendig med en enkel oppdatering av databaseversjonen, eventuelt kunne databasen «hostes» på en Linux distribusjon, da SNI-feilen ikke eksisterte på Linux-versjonen av Oracle Database 11g. Dermed fikk utviklerene først migrert «workspacet» til en Linux «hostet» database, men oppdragsgiver oppdaterte allikevel til nyeste versjon av databasen. Den nye versjonen, 12.2.0.1 (12c R2), fikser som sagt SNI, og systemet kunne da like godt kjørt på Windows.

7.2.3 Sprint 3

Periode: 26.02 - 09.03
Total tidsbruk: 187 timer
I Sprint 3 var fokuset å fullføre kartleggingen av informasjon slik at moddelering og oppsett av databasen kunne bli ferdig. På dette tidspunktet hadde utviklerteamet ganske god kunnskap om hva som kunne være verdifull informasjon for oppdragsgiver, og hadde derfor et mer detaljert kravspesifikasjonsmøte(se vedlegg H.2.5). Denne sprinten representerte en «milepæl» for utviklerteamet da kommunikasjonen med API-et endelig var etablert og teamet kunne bekrefte at systemet faktisk kunne utvikles. Nå kunne implementering og programmering av den faktiske løsningen starte.

API

Teamet fikk også autentisert mot API-et fra APEX ved bruk av pakken «utl_http». Den fungerer da med en følgende spørring:

```
utl_http.set_wallet(p_wallet_path, p_wallet_password);
```

En egen wallet mot Aruba Central måtte opprettes, og dette ble gjort i:

```
file:c:/oracle/wallet_aruba
```

Teamet opprettet da en en systemtabell, AWS_SYSTEM_PARAMS, som blant annet inneholder en rad med ulike parametere som benyttes i systemet, for eksempel kryptert passord, client id/client secret, osv.

Rammeverket og autentisering skulle nå være i orden, og teamet testet noen av spørringene og fikk korrekt svar. Allikevel var det fremdeles problemer med å få den riktige informasjonen fra API-et for alle ønskede spørringer. Det viste seg at det ikke var nok å aktivere API-Gateway for kunden, siden informasjonen fra API-et fortsatt ikke kunne sammenlignes med forventede resultater for noen av spørringene. Først antok teamet at en MSP-bruker kunne utføre «monitoring»-spørringer på alle kunder, men det viste seg at dette ikke stemte. Teamet kom da frem til at det var nødvendig å autentisere seg for hver enkelt kunde og gjøre monitoring-spørringer hver for seg mot API-et.

Dette medførte da store endringer i hvordan uthenting av informasjon initielt var planlagt, og lagring av denne. Teamet brukte her innloggingsakkreditiver (client id og client secret) for hver enkelt kunde, til å autentisere og lage tokens. Videre måtte disse tokens lagres for hver enkelt kunde. Da dette ble implementert, hadde systemet fungerende og riktig tilgang/autentisering. Teamet kunne nå starte behandling av responsen til URL-ene som var kartlagt tidligere.

Database

Databasen ble først modulert med utgangspunkt i svareksemplene som ble vist i API-Gateway, men ble noe kuttet ned etter kravspessifikasjonsmøtet, da enkelte tabeller ble overfladiske. I denne sprinten ble det også testet å

legge inn data fra API-et inn i en tabell, og vise data fra database i webapplikasjonen. Implementasjon av kryptering ble også påbegynt denne sprinten.

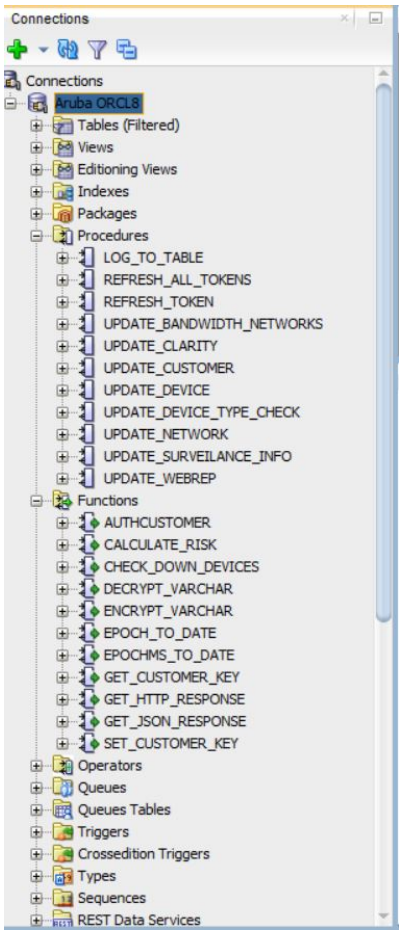
7.2.4 Sprint 4

Periode: 12.03 - 23.03
Total tidsbruk: 172 timer

I sprint 4 var fokuset å hente ut informasjon fra API-et, og lagre dette i databasen. Dette var den første sprinten der fokuset fullt og holdent innebar utvikling av systemet. Den initielle tiden satt av til utvikling var nå blitt mindre, mye grunnet nødvendige foregående prosesser og utfordringer. Teamet ønsket å holde god standard og var bevisste på kodekvalitet i systemets kode, også med tanke på at det erfaringsmessig vil komme flere utfordringer. Videre i sprinten ble krypteringsalternativer i Oracle Database drøftet, en «enkel» krypteringsmetode ble tatt i bruk.

Mye av tiden gikk med på funksjoner som skal hente data fra API om nettverk, nettverkshelse og enhet, og sette inn/oppdatere tabellene med disse data.

På slutten av denne sprinten fungerte samspillet mellom AWS_CUSTOMER_DETAIL, AWS_CUSTOMER og AWS_CUSTOMER_KEY. Dette vil si at brukeren kan legge til en kunde i GUI, for så å bli lagt inn i riktig tabell i databasen. Når denne tabellen blir oppdatert, enten ved UPDATE- eller INSERT-utsagn, blir en «trigger», oppretter eller oppdaterer deler av innholdet i en gitt rad. Videre ble kunden autentisert, og korrekte variabler som tokens ble lagt inn i sine



Figur 7.1: Skjermbilde av filstruktur i SQL Developer

respektive tabeller. Store deler av filene som
vist i figur 7.1 var nå på plass.

7.2.5 Sprint 5

Periode: 03.04 - 13.04
Total tidsbruk: 204 timer

I sprint 5 var fokuset på å ferdigstille løsningen i størst mulig grad. Etter utviklernes påskeferie fungerte ikke access tokens for kundene. Årsaken til den nye feilen var rettighetsendringer i forhold til MSP.

Endringer av rettigheter

Slik utviklerne hadde implementert autentiseringen før påske, var å bruke client id og client secret til MSP for autentisering av alle kunder tilknyttet dennne kontoen. Kundene ville da få en access token som kunne brukes i alle spørringer mot egne nettverk. Teamet sendte en e-post til Aruba TAC vedrørende disse rettighetsendringene, da det viste seg som vanskelig å finne informasjon om dette i «change log». Det kom da frem at hver enkelt kunde nå måtte ha hver sine sett med client id og client secret. Teamet kunne da fastsette at access token generert ved hjelp av MSP-akkreditiv ikke lenger kunne brukes i sammenheng med andre kunder, slik som tidligere.

Konsekvenser

Som et resultat av dette måtte kundene nå autentiseres med sine respektive akkreditiv. Dette medførte at disse først måtte opprettes i Aruba Central. Videre må disse legges inn i brukergrensesnittet ved opprettelse av en kunde, for så å lagre dem i databasen for hver av kundene. Dette medførte store endringer i koden som måtte implementeres, samtidig som det var nødvendig å rette ett større fokus på å bli ferdig med brukergrensesnittet. Hovedsakelig var jobben her å lage korrekte SQL-spørringer for å hente ut aktuell data fra tabellene, da APEX vil ta hånd om visningene selv.

Videre i sprinten

Videre bestemte utviklerne, i samråd med oppdragsgiver, å bruke APEX sin «plug in» for autentisering inn mot webapplikasjonen.

Resten av tiden ble brukt på å refaktorisere og automatisere funksjoner og prosedyrer. Prosedyren UPDATE_SURVEILLANCE ble implementert som en hovedfunksjon hvor alle andre «update»-funksjoner ble initiert fra. Denne ble satt opp som en scheduled job og var fungerende ved slutten av denne sprinten.

7.2.6 Sprint 6

Periode: 16.04 - 27.04
Total tidsbruk: 192 timer

I Sprint 6 var fokuset å ferdigstille løsningen totalt, og skrive rapport. For å håndtere feil i systemet ble det implementert logging av disse til tabell. På den måten kunne overvåkningsteamet gjøre enkel feilsøking av systemet selv.

Siste del av UPDATE_NETWORK ble fullført og en ny jobb som henter daglig informasjon om mengde nettverkstrafikk ble implementert. Webapplikasjonen ble internasjonalisert fra engelsk til norsk, gjennom funksjonalitet i APEX. APEX sin egen «Advisor» ble kjørt for å sjekke løsningen på blant annet «best practice» og sikkerhet. Resultatet viste mest «grønt lys», men gjorde utviklerene oppmerksom på at alle sidene i applikasjonen var under standard autentisering. Dette var ikke et problem, på grunn av at systemet kun inneholder ett brukernivå som skal ha tilgang til alle sider i webapplikasjonen. Det ble også gjennomført en akseptansetest av løsningen og oppdragsgiver var godt fornøyd med produktet.

I forhold til GUI ble hovedvisningen «Overview» ferdigstillt, da med innhold i forhold til kravene. Det er også lagt inn en automatisk oppdatering av denne siden, med et intervall på fem minutter.

7.3 Ferdigstilling av rapport

Periode: 30.04 - 15.05
Total tidsbruk: 355 timer

I tiden etter prosjektets slutt ble rapporten hovedfokus, og all utvikling stanset, utenom noen mindre «bug fixes». Gruppen hadde allerede skrevet deler av rapporten ved siden av utviklingen, men det var nå helheten skulle ferdigstilles. Under skrivingen har de ulike gruppemedlemmene tatt for seg forskjellige deler og tema i rapporten. Videre har prinsipper som «parprogrammering» blitt brukt i form av at en annen i gruppen alltid leser over de andres arbeid.

Videre hadde gruppen en avtale med en annen bachelorgruppe ved NTNU i Gjøvik, hvor rapporter ble utvekslet, og tilbakemeldinger gitt. Dette var en nyttig prosess for å få et annet perspektiv på rapporten, da hovedsakelig dens innhold og struktur.

8 Avslutning

Oppdragsgiver hadde fra begynnelsen av prosjektperioden uttrykt at utviklerteamet sto med mye frihet med tanke på løsning og utviklingsmetodikk. Dette åpnet for muligheten til å konkretisere og påvirke mange av kravene spesifisert i kapittel 3 selv. Dette medførte en lærerrik prosess hvor samtlige gruppemedlemmer fikk mulighet til å uttøve sine kreative evner og logiske tankegang for å lage en så praktisk og brukervennlig løsning som mulig.

Utviklerne har ved flere anledninger blitt overrasket over hvor ofte teorier, prinsipper, og metoder fra det som er blitt undervist i nesten samtlige fag gjennom de siste årene, har kommet til nytte i løpet av prosjektperioden. For eksempel prinsipper og metoder innenfor databasebehandling ble viktig å forstå, for å lage en god grunnmur til løsningen. Teamet forsto raskt hvor viktig de grunnleggende programmeringsferidghetene er, og hvor viktig det ble å ha en felles forståelse av kvalitet og standarder.

8.1 Resultat

Produktet bærer preg av en vel gjennomført kommunikasjonsprosess mellom utviklere og oppdragsgiver. Webapplikasjonen ser ikke særlig komplisert ut fra et utsideperspektiv, noe som var intensjonen helt fra begynnelsen av. Det eksisterer allikevel en omfattende mengde med støttefunksjonalitet, spesielt i forhold til databasen. Dette innebærer i stor grad informasjonsinnhenting, oppdatering og vedlikehold av data i databasen, i tillegg til å sørge for at kommunikasjonen mellom systemet og API-et blir opprettholdt.

Å få kontakt, hente og prosessere informasjon fra API-et var en oppgave som oppdragsgiver beskrev som utfordrende. Utviklerteamet er meget fornøyd med at dette ble oppnådd på mindre tid enn i verst tenkelig scenario.

Både utviklerne og oppdragsgiver er svært tilfreds med resultatet av dette prosjektet, begge parter har konkludert med at styringssystemet er i henhold til de prosjektmålene som ble satt. Styringssystemet har allerede skapet merverdi for oppdragsgiver som leverandør av NaaS.

Kravspesifikasjonen har i møter[1] undervegs i utviklingen blitt spesifisert og endret på, samtidig som teamet har kommet med innspill til løsningen. For eksempel er det vurdert hvilke parametere som skal overvåkes i løsningen gjennom kartlegging av disse, og fremmet forslag om hvilke som kan være interessante for oppdragsgiver. Systemet har i tillegg egne respektive oversikter over alle nettverk og enheter. Dette var initielt ikke et krav, men det kom som et følge av en grundig, strukturert database der informasjonen om nettverk og enheter ble fremstilt på en ryddig og informativ måte. Databasen åpnet med andre ord opp for ekstra funksjonalitet som ble vurdert som verdifull, selv om det i utgangspunktet ikke var tenkt på.

Kort oppsummert fungerer systemet i dag på følgende måte: Etter at en bruker har logget inn, vil systemet vise all relevant informasjon om en kunde, med tilhørende nettverk og enheter i webapplikasjonen. Informasjonen i oversikten oppdateres kontinuerlig, samt at innholdet kan sorteres på den måten overvåkningsteamet anser som hensiktsmessig.

8.2 Vurdering av sikkerheten

Oppdragsgiver ytret et ønske om at sikkerhet og eventuelle tiltak i løsningen skulle vurderes etter beste evne. Dette så gruppen på som en god utfordring da fagområdet er spennende og aktuelt for alle som ønsker å jobbe med informasjonsteknologi. Dette har vært et fokus gjennom hele prosjektet, der for eksempel databasen ble modellert med hensyn på sikkerhet, og da spesielt med tanke på hvordan informasjon skulle krypteres.

Sikkerhet opp mot Aruba Central API-et ble også satt i fokus. Som tidligere nevnt i 5.2.3 måtte sertifikatet til Aruba Central legges inn i databasens wallet for at kommunikasjonen mot Aruba skulle godkjennes. Utviklerne gjorde seg oppmerksomme på at sertifikatet var utstedt av GeoTrust, eid av Symantec, og førte en dialog med oppdragsgiver og veileder rundt dette. Veileder poengterte at Symantec ikke lenger er ansett som «tillitsfull» sertifikatutsteder, noe som høyst sannsynlig ville resultere i at Aruba kom til å bytte sertifikat en gang i nær fremtid. Dette inntraff den 04.05.18, da ingen kommunikasjon mot API-et lenger fungerte. Da utviklerne sjekket sertifikatene

[1] se Møtereferat: Oppdragsgiver i vedlegg H

viste det seg at Aruba hadde byttet sertifikatutsteder, slik som tidligere antatt. Det nye sertifikatet måtte da hentes fra Aruba Centrals systemer, og legges inn på ny i wallet. Selv om systemet da hadde vært nede i over et døgn, re-autentiserte det seg på korrekt måte, og gikk tilbake i operasjonell tilstand, uten innblanding fra utviklerne.

8.3 Alternativer

Tidlig i utviklingsprosessen sto utviklerne overfor et veivalg(2.3) mellom å følge den initielle prosjektplanen med Node.js som plattform, eller gå for den foreslåtte Oracle Database- og APEX-plattformen. I begynnelsen var utviklerne bekymret for at APEX-alternativet skulle ta bort mye av læringsmulighetene ved prosjektet, og tenkte at dette valget ville medføre at mye funksjonalitet som teamet selv kunne sette seg inn i og utvikle, i stedet ville løses av «ferdiglaget» funksjonalitet.

Det initielle alternativet ville ført til at teamet selv måtte ha satt opp en egen server med Node.js, samt utvikle mye mer for å oppnå ønsket funksjonalitet rundt for eksempel brukergrensesnittet. Denne plattformen ble lenge vurdert på grunn av utbyttet, i form av erfaring med moderne og velkjente rammeverk for web-utvikling. Men på grunn av potensiell tidsbruk, en noe mer omfattende database enn først tenkt, og ikke minst hvor lettvint det ville være for oppdragsgiver å videreutvikle, falt valget på Oracle-plattformen. Teamet hadde også initielt planer om å implementere grundigere testing av all funksjonalitet i løsningen, men her tar teamet selvkritikk på at dette ikke ble gjennomført i like stor grad som planlagt. Å utføre denne typen testing ble meget tidkrevende, og teamet prioriterte ikke dette like høyt som god praksis tilsier at det burde.

Initielt designet teamet løsningen etter «Observer pattern»(seksjon 4.3.2), dette er et godt mønster som beskrev den planlagte løsningen på en god måte. Da teamet skulle implementere varslingen fra Subjekt til Observatør om at det er skjedd en oppdatering, rådførte teamet seg med en ansatt hos oppdragsgiver. Der ble det konkludert med at den mest effektive måten å gjøre dette på var å oppdatere GUI hvert femte minutt, uavhengig av oppdatering av databasen (Subjekt). Dette avviker noe fra mønsterets intensjon, men er en god løsning for systemet.

8.4 Kritikk av oppgaven

Oppgaven presentert i november appellerte til alle gruppemedlemmene, og den virket engasjerende å arbeide med. En større utfordring var at det ikke var definert kravspesifikasjon før mot slutten av februar. Fram til dette kravspesifikasjonsmøtet(H.2.5) var det en del uklarheter rundt hva oppdragsgiver egentlig ville ha av informasjon vist i overvåkingssystemet. Utviklerne laget seg en oversikt over hva som fantes av informasjon, men ikke hva som var viktig for oppdragsgiver. Dette betyr på ingen måte at teamet ikke jobbet målrettet med utvikling i januar og februar, men at denne tiden kunne bli utnyttet bedre. Utfordringene i sammenheng med API-et var også større enn utviklerne så for seg. Prosessen rundt å identifisere, utforske og implementere problemene var lærerikt, men teamet kunne gjerne sett for seg å ha hatt et større fokus på utvikling, og da spesielt programmering.

8.5 Evaluering av arbeidsprosess

Flere ganger gjennom prosessen har utviklerne hatt problemer med å opprettholde kommunikasjon mot API-et. Utviklerne har da vært flinke til å jobbe med andre oppgaver mens de ventet på svar fra aktuelle aktører, noe som medførte at utviklingens gang ikke ble svekket i kritisk grad. Teameet har sett verdien av en grundig risikoanalyse og fokus på tiltak i praksis.

Det å lære seg et nytt programmeringsspråk(PL/SQL) gikk greit for alle utviklerne. Dette tyder på en god grunnforståelse i programmering generelt, som samtlige utviklere har fått gjennom studiet.

I startfasen av prosjektet ble det brukt mye tid på å sette seg inn i JavaScript og tilhørende rammevert, før det ble bestemt at rammeverk skulle byttes. Dette gjorde at utviklerne i praksis lærte seg to forskjellige språk og tilhørende rammeverk, som begge var relativt nye for utviklerne. Dette medførte at tid, som kunne bli brukt til utvikling og problemløsning, gikk med til ting som ikke ble «nyttige» gjennom prosjektet, men som har gitt erfaring å ta med videre.

Prinsippet On-site customer fungerte meget bra. Dette i form av forskjellige ansatte hos oppdragsgiver, som utviklerne hadde mulighet til å prate med

vær tirsdag og torsdag da de var på oppdragsgivers kontorer. Dette har ført til tett dialog, som igjen har forhindret misforståelser som kunne ha oppstått over for eksempel epost og bygget opp under en smidig utviklingsprosess.

8.5.1 Organisering

Siden ingen av utviklerne hadde noen praktisk erfaring med Scrum fikk samtlige gjennom utviklingens gang god læring i å holde «daily scrum», «sprint planning meeting» og «sprint retrospective meeting». Verdien i disse møtene kom fort fram ved at de ga innsyn i hvordan ting hadde gått så langt i utviklingen hva som må gjøres nå, og hva som skal gjøres framover. Utviklerne synes det å bruke Scrum som utviklingsmodel har fungert veldig bra i forhold til oppgaven. Fordelen med å bruke smidig utvikling kom godt fram en måned inn i utviklingen da oppdragsgiver kom med mer spesifikke krav til overvåkingssystemet. Nye user stories i henhold til disse tilspissede kravene ble lagt til i backlog og tatt med i neste sprint.

8.5.2 Prosjekt som arbeidsform

Gjennom prosjektperioden har gruppen opparbeidet seg erfaring i det å arbeide med et større prosjekt, og i et team hvor alle har vært avhengig av hverandre for å skape et produkt med høy nytteverdi. Det har vært engasjerende å arbeide med noe som faktisk skal tas i bruk, og ikke bare som innlevering til sensur i et emne ved skolen. På denne måten har gruppen vært avhengig av å tilrettelegge utviklingen for å tilfredstille krav formulert av en oppdragsgiver, og ikke bare utvikle etter egne preferanser og standarder, slik man gjerne gjør i et selvstendig arbeid i skolesammenheng. Det har vært en positiv opplevelse for samtlige gruppemedlemmer å jobbe i team, kontra det å jobbe selvstendig, da medlemmene hele veien har kunnet spille på hverandres kompetanse og erfaringer.

Oppsummert sitter gruppen igjen med et helhetsinntrykk av at dette prosjektet har vært en særdeles lærerik, virkelighetsnært, en meget relevant prosess, og en god erfaring å ta med seg videre til fremtidige prosjekt av liknende karakter i arbeidslivet.

8.6 Videreutvikling

Oppdragsgiver hadde mange idéer rundt hva overvåkingssystemet kunne vise av informasjon. Noen av planene ble ikke gjennomført nå i første omgang på grunn av begrenset mengde med tid og utviklere, og er derfor beskrevet kort her:

Validering av input når brukeren legger til en ny kunde må ferdigstilles. Utviklerne begynte med validering på deler av input, men det ble nedprioritert for å ferdigstille viktigere deler av systemet.

Grunnen til at det har blitt brukt en database som grunnmur er for å blant annet kunne generere historikk for å kunne avdekke trender. Nå ligger den kontinuerlig oppdaterte informasjonen i oversiktlige tabeller der informasjon det ønskes å føre historikk på enkelt kan hentes ut med en SELECT-statement. For å lage faktisk historikk må det implementeres enkle nye tabeller som holder en instans av en tabell med et gitt tidspunkt. En «trigger» som lagrer denne historikken ved UPDATE- og/eller INSERT-utsagn kan bruker for å lagre aktuell tabelldata i den nye «historikktabellen».

I koden så behandles unntak, men de blir stort sett bare logget. Her kan det forbedres ved å ha spesifikke unntaksbehandlinger ved for eksempel «no data found» ved uthenting av krypteringsnøkler til en kunde.

Når nye kunder blir lagt til systemet, finnes det ingen måte å fjerne disse, uten å bruke databasen direkte. Slik funksjonalitet kan oppnås gjennom APEX, men ble utelatt på grunn av sikkerhetsvurderingene som ble gjort. På denne måten vil ikke webapplikasjonen kunne slette utilsiktet data, som kan ha følger for systemet. Om en kunde er lagt inn feil, finnes muligheter for editering i systemet. Dette er noe som bør vurderes ved en eventuell ny utviklingsperiode.

Utviklerteamet startet også henting av innformasjon innenfor WEB_REPUTATION, men fikk problemer da innholdet i spørringen varierte i for stor grad. Tabellen skulle inneholde tallmateriale som er aktuelle for oppdragsgiver. Kildekode for, og i sammenheng med WEB_REPUTATION, inkludert tabellen, er på bakgrunn av dette beholdt, slik at de som skal videreutvikle løsningen kan fortsette på det som allerede er kodet. Her var intensjonen å lagre hver enkel type nettverkstraffikk, per nettverk, i samme tabell. Altså vil det for eksem-

pel kunne finnes 6 kolonner per nettverk, med de forskjellige variablene.

8.7 Konklusjon

Når teamet nå ser tilbake på utviklingsperioden, kan det refereres til gode erfaringer fra utfordrende prosesser og lærerike arbeidsoppgaver.

Teamet har fått et bredere erfaringsgrunnlag rundt det å være dataingeniør, og har både gjennom utvikling og rapportskriving sett og forstått sammenhenger mellom ulike fagområder. Gruppen føler nå at de går ut i arbeidslivet med en vesentlig bedre forståelse for at det fortsatt er mye å lære!

Bibliografi

[1] BIG THINK EDITORS. Big idea: Technology grows exponentially. http://bigthink.com/think-tank/big-idea-technology-grows-exponentially. lastet 02.05.2018.

[2] Big Data BBVA. Eight reasons to use 'cloud computing' in the company. https://www.bbva.com/en/eight-reasons-to-use-cloud-computing-in-the-company/. lastet 01.05.2018.

[3] Wikipedia. Nettskyen — wikipedia,, 2018. [På internett; besøkt 23-april-2018].

[4] EVRY Norge AS. Network as a service. https://www.evry.com/no/bransjer-og-tjenester/tjenester/infrastrukturtjenester/sikkerhet/network-as-a-service/.

[5] Oracle. Oracle application express, deployment. http://www.oracle.com/technetwork/developer-tools/apex/application-express/apex-deploy-installation-1878444.html. lastet 02.05.2018.

[6] Oracle. Oracle database and pl/sql. https://www.oracle.com/tools/solutions.html#database. lastet 04.05.2018.

[7] Mark Richards. Software architecture patterns. https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html. lastet 04.05.2018.

[8] Erich Gamma; Richard Helm; Ralph Johnson; John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, October 31, 1994.

[9] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, October 2004.

[10] Oracle. Building and customizing an interactive grid in oracle application express 5.1. http://www.oracle.com/webfolder/technetwork/tutorials/obe/db/apex/r51/building_and_

customizing_interactive_grid/interactive_grids_part1.html. lastet 04.05.2018.

[11] Oracle. Timesten in-memory database sql reference; privileges. https://docs.oracle.com/cd/E11882_01/timesten.112/e21642/privileges.htm#TTSQL338. lastet 30.04.2018.

[12] Oracle. Dbms_crypto. https://docs.oracle.com/database/121/ARPLS/d_crypto.htm#ARPLS664. lastet 09.05.2018.

[13] Oracle. Dbms_crypto, randombytes function. https://docs.oracle.com/database/121/ARPLS/d_crypto.htm#ARPLS65704. lastet 19.04.2018.

[14] Datatilsynet. Hva er en personopplysning? https://www.datatilsynet.no/om-personvern/personopplysninger/. lastet 19.04.2018.

Vedlegg

A Prosjektavtale

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

EVRY NORGE AS (oppdragsgiver), og

Snorre Rogne, Guro Storlien,
Bjørn Ole Myrold, Sondre Ahlgren

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 1/1-18 til 16/05-18.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Eivind Alnestad

Oppdragsgivers kontaktperson (navn): Bjørn Kåre

Student(er) (signatur): Guro Storlien dato 9/1-18

Snorre Ragne dato 09/01-18

Sondre Ahlgren dato 9/1-18

Bjørn Ole Myrold dato 09.01.18

Oppdragsgiver (signatur): Bjørn Kåre dato 9/1-18

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

B Oppgavebeskrivelse

Dato: 25.10.17

evry.com

Oppgave

WiFi as a Service (WaaS)

Bachelor oppgave innen IT

Innhold

1 Innledning

1.1 Bakgrunn

Stadige flere produkter leveres i dag som tjenester. Dette muliggjøres gjennom sentrale og konsoliderte plattformer som er samlokalisert og forvaltet i regi av produsent. For mange av disse løsninger brukes begrepet «Cloud». Gjennom «Cloud» eller skybaserte løsninger gis brukere mulighet til å få tilgang til og dele for eksempel datanettverk, servere, lagring, applikasjoner og tjenester.

1.2 Fagområde

Utviklingen innen skybaserte løsninger har akselerert de siste årene. Vi kjenner tidlig til applikasjoner som presenteres og leveres over web grensesnitt. De siste årene har infrastruktur for datatrafikk og nettverk gjort det mulig å i større grad levere tjenester innen IaaS (Infrastructure as a Service) og PaaS (Plattform as a Service).

Infrastruktur nettverk og barriere sikringer (brannmur og tilsvarende) er fortsatt sentrale komponenter for lokal IT forvaltning. Dog gir brukeres krav til mobilitet og tilgjengelighet utfordringer for IT, og systemene blir mer komplekse med økende krav til sikkerhet.

1.3 Trådløse nettverk

Det vil her være fordelaktige med tanke på fremtidig drift og løpende kostnader, er å se på en skybasert løsning innen området. Løsninger i dag vil gi mulighet for å fjerne den lokale aksesspunktkontroller som skal vedlikeholdes og administreres, og heller da at all sentral funksjonalitet leies som en løpende tjeneste i skyen fra ulike produsentene.

2 Oppgavebeskrivelse

2.1 Tittel

Hvordan kan Aruba sin skybaserte løsning tilpasses slik at den på en oversiktlig måte gjennom management imøtekommer kundens krav til SLA (Sevice level agreement).

1. Vurdere hvilke parameter som skal overvåkes i løsningen
2. Tilpasse parameter ut fra tilgjengelige API sett
3. Vurdere sikkerhet og eventuelle tiltak i løsningen

2.2 Oppdragsgiver

Oppdragsgiver er:

EVRY Norge AS
Strandveien 2
2360 Brumunddal

2.3 Kontaktperson

Bjørn Kråkevik
Mob: 97071327
e-post: bjorn.krakevik@evry.com

2.4 Beskrivelse av oppgaven/ oppdraget

Se pkt 2.1 samt utfyllende opplysninger nedenfor i systembeskrivelse.

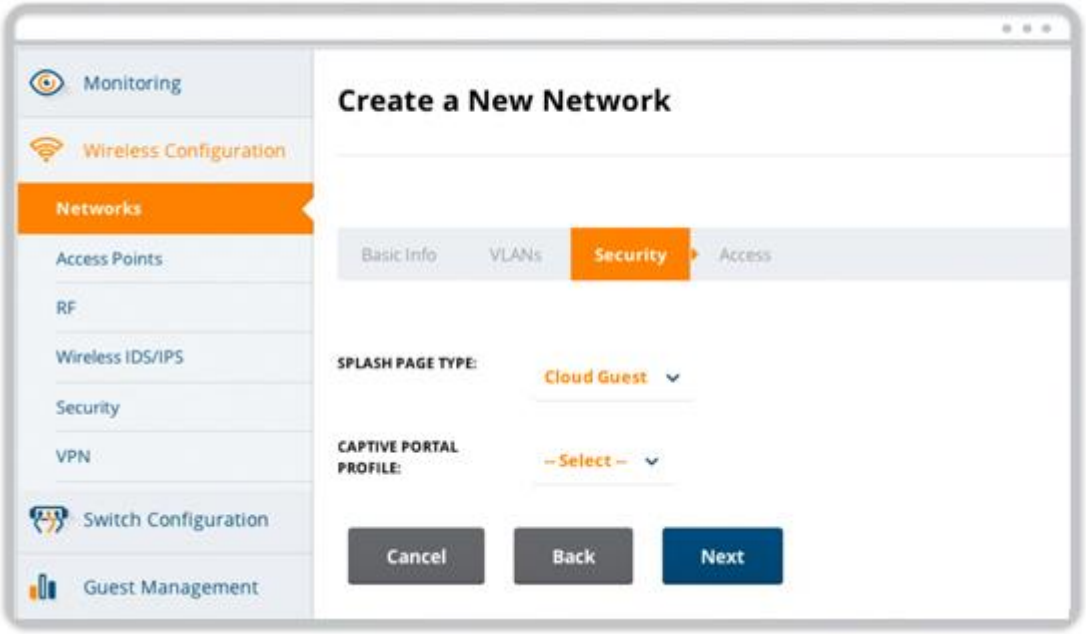
Oppsummert ønsker vi at studentene tar utgangspunkt i dagens løsning som er i produksjon. Dette er en «multitenant» løsning med skybasert management. Managementet er i dag ikke tilpasset det som support personell har behov for å se ut ifra den SLA som er tegnet med kunder. Det finnes et sett med API som gjør det mulig å tilpasse managementet. Vi ønsker at studentene vurderer hvilke API sett som gir best oversikt i samme bilde og legger disse opp tilpasset en storskjerms visning. Samtidig ønskes det at de setter seg inn i teknologien og gir

3 Vedlegg 1 : Skybasert løsning fra Aruba

Løsningen krever ikke egen kontroller. Løsningen er basert på Aruba Instant 802.11ac AP'er. Aruba Instant har virtuell kontroller innebygget i AP'ene som styres med skybasert management, Aruba Central.

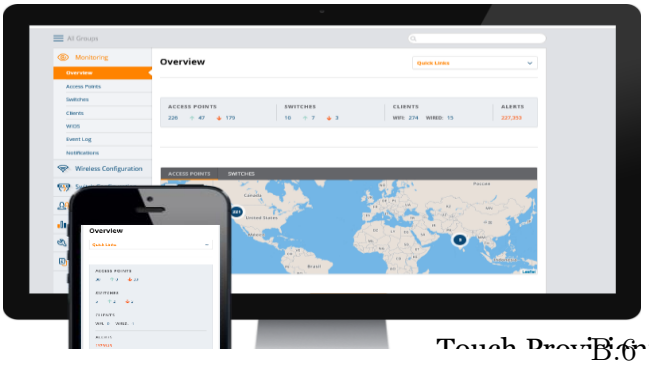
Aruba Central som nevnt ett skybasert system for å håndtere både trådløse og kablede nettverksenheter på en enkel måte.

Det grafiske brukergrensesnittet er intuitivt og bygget opp slik at det ikke er behov for mye kompetanse for å konfigurere og drifte utstyret.



Løsningen støtter mange muligheter for gjestenett, det kan være lokale brukere, integrasjon med 3.partsløsning, innlogging med sosiale nettverk som for eksempel Facebook, Google+, LinkedIn, SMS, epost bekreftelser mm.

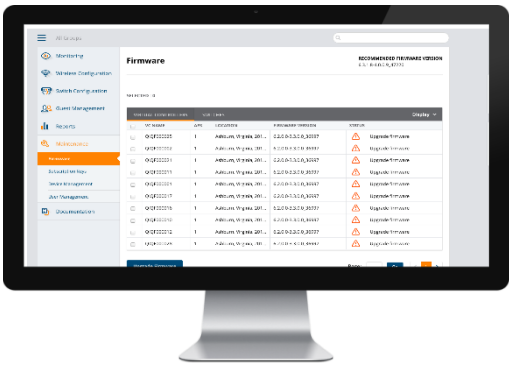
En viktig fordel med Aruba Central er at det lokale nettverket vil fungere også om internettforbindelsen skulle gå ned siden selve konfigurasjonen ligger lagret lokalt. Det er også mulig å hente se på relevante informasjon om AP'er, brukre o.l. lokalt (i tillegg til se det i Central). Den eneste konsekvensen av å miste internettforbindelsen er at man ikke lenger kan logge på skytjenesten for å hente ut informasjon der.



Aruba Central tilbyr også ZTP – Zero Touch Provisioning slik at man kan gjøre klart

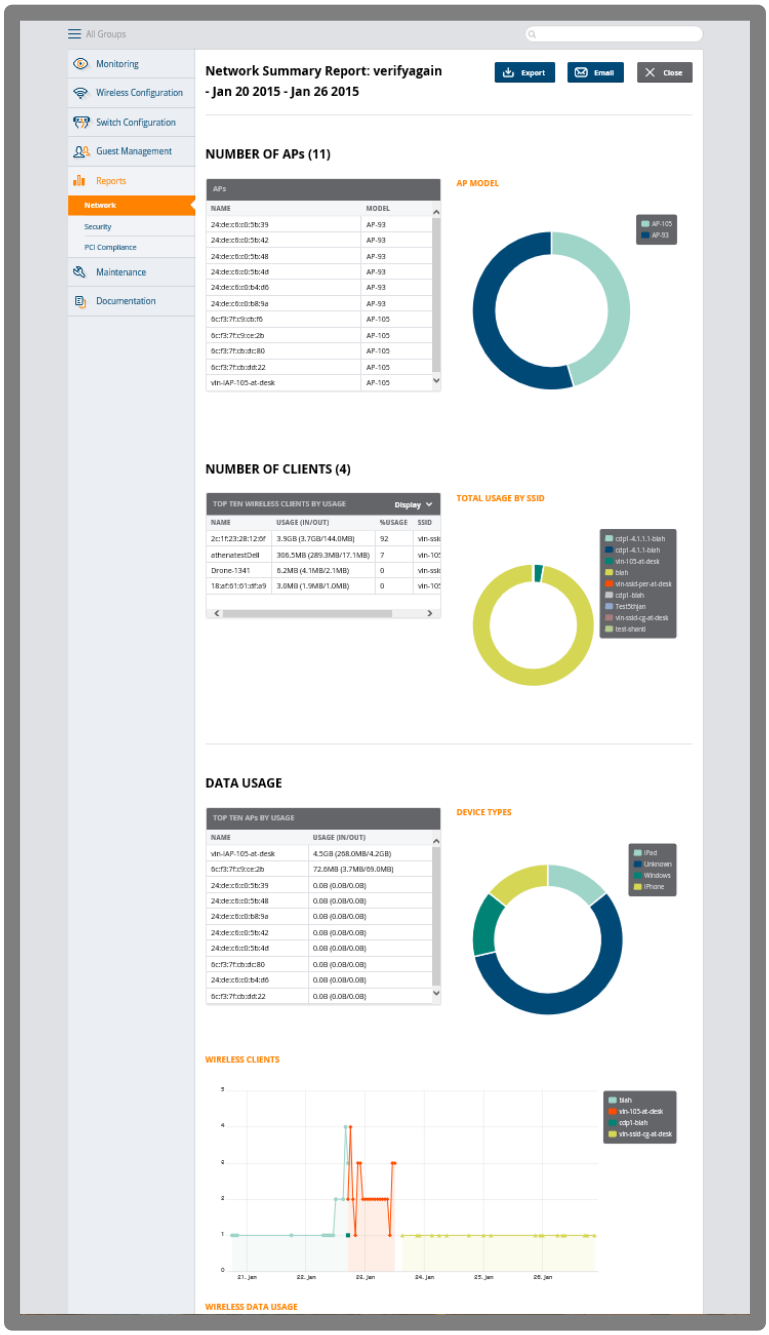
Det må nevnes at i motsetning til konkurrerende løsninger, så kan dette utstyret konverteres til å ha lokal managment,eller brukes med dedikert hardware kontroller om behovene skulle endre seg over tid. Dette gjør at man kan endre behov over tid, uten at man må kjøpe ny AP'er og bruke mye ressurser på å bytte dem. Aruba har også overlegen støtte for AirGroup hvor man enkelt kan velge hvilken type tjenester som skal tilbys brukerne. Dette kan skreddersys ytterliggere om man også benytter dette med Aruba ClearPass (opsjon).

Det er enkelt å lage grupper for lokasjoner som skal ha samme grunnoppsett, men kanskje noen lokale tilpasninger. Dette gjøres enkelt ved at man lager ulike grupper for eksempel per lokasjon. Konfigurasjonen kan deretter kopieres eller flyttes mellom de ulike gruppene og så kan man eventuelt gjøre ønskede tilpasninger for ulike grupper.



Aruba Central har gode muligheter for å definere ulike parametere for IDS/IPS og brannmurregler basert på ulike roller og det er enkelt å se status på AP'er, oppe/nede/antall.

Aruba Central har også en egen app for smarttelefon/nettbrett som gjør det enkelt å følge med på hva som skjer i nettverket, uten å måtte logge seg på noen PC.



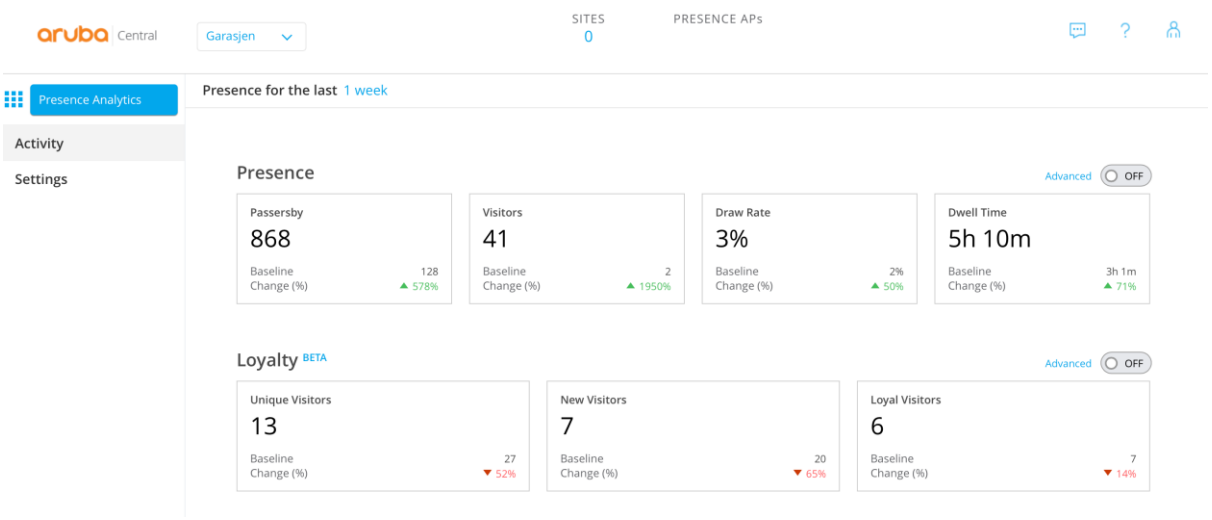
Gjestenett-løsning

Det er enkelt å lage tilpasninger for ulike enheter, lokasjoner o.l. Selv-registering er også mulig, med valg for epost og SMS. Også dette er lisensiert per AP, ikke bruker, og benytter ett token per AP.

Aruba Central/Instant støtter kommunikasjon direkte mot LDAP, radius og Tacacs. Brukes i de tilfellene som kunder har egen elektronisk gjestebok med mulighet for å integrere mot 3-part.

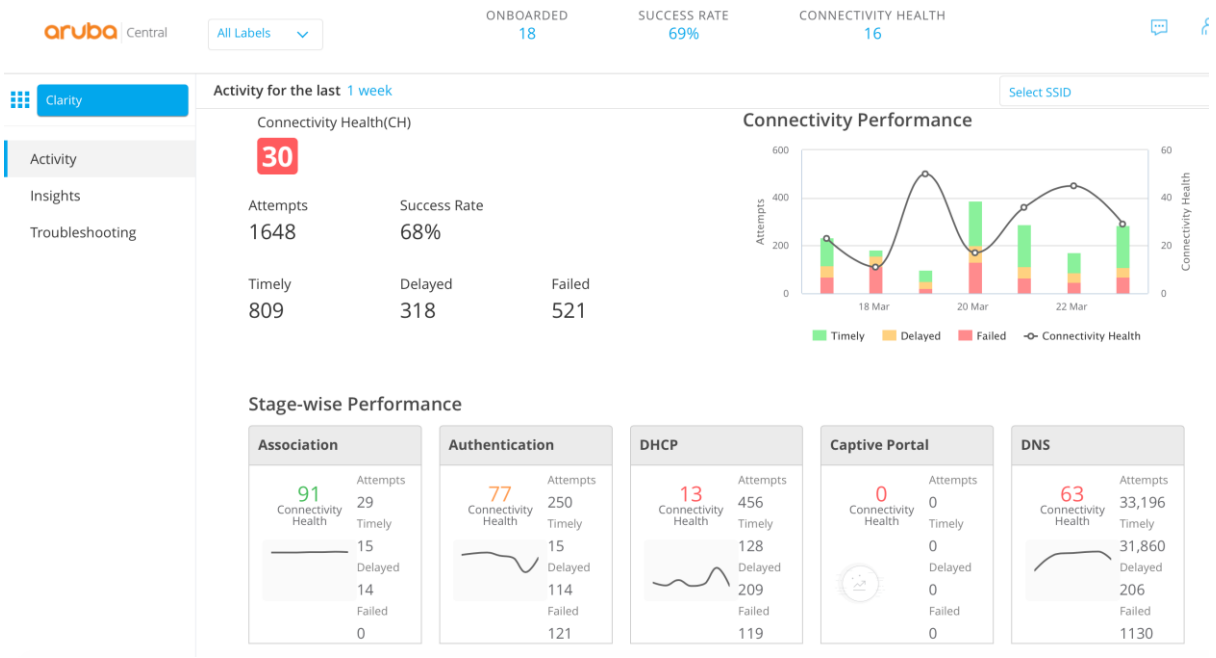
Presence Analytics

Det er også mulig å utvide funksjonaliteten med Presence Analytics (benytter ett token per AP som skal ha funksjonaliteten). Ved å benytte denne delen i Aruba Central, så kan man få innsyn i antall besøkende i nærheten av aksesspunktet. Man kan feks også innsyn få i hvor lenge en person har vært i butikken mm.



Nye funksjoner

Aruba Central står sentralt i Aruba sin satsning fremover. Dagens funksjoner blir stadig utvidet, og i tillegg kommer det nye funksjoner. Pt. er den svært godt mottatte funksjonen Clarity i Central. Clarity gir innsikt i viktige funksjoner som ikke nødvendigvis direkte har noe med nettverket å gjøre, men som er viktig for at **tjenesten nettverk** skal fungere tilfredsstillende. Eksempler er overvåkning av DNS og DHCP tjenester. Det har vist seg at mange av de diffuse problemene som trådløse brukere sliter med har sin årsak i problemer med denne type tjenester.


aruba Central
All Labels
ONBOARDED
18
SUCCESS RATE
69%
CONNECTIVITY HEALTH
16
Clarity
Activity
Insights
Troubleshooting
Activity for the last 1 week
Select SSID
Connectivity Health(CH)
30
Attempts
1648
Success Rate
68%
Timely
809
Delayed
318
Failed
521
Connectivity Performance
Attempts
Connectivity Health
18 Mar
20 Mar
22 Mar
Timely
Delayed
Failed
Connectivity Health
Stage-wise Performance
Association
91
Connectivity Health
Attempts
29
Timely
15
Delayed
14
Failed
0
Authentication
77
Connectivity Health
Attempts
250
Timely
15
Delayed
114
Failed
121
DHCP
13
Connectivity Health
Attempts
456
Timely
128
Delayed
209
Failed
119
Captive Portal
0
Connectivity Health
Attempts
0
Timely
0
Delayed
0
Failed
0
DNS
63
Connectivity Health
Attempts
33,196
Timely
31,860
Delayed
206
Failed
1130

C Prosjektplan

Prosjektplan - NaaS

BIDAT39 - Bacheloroppgave Dataingeniør

Bjørn Ole Myrold
Sondre Ahlgren
Snorre Rogne
Guro Storlien

Institutt for datateknologi og informatikk NTNU
Gjøvik, 16. mai 2018

C.1 MÅL OG RAMMER

C.1.1 Bakgrunn

Dette prosjektet kommer som en følge av Bachelorprogrammet ved NTNU i Gjøvik, der studenter ved universitet skal utføre en større oppgave, gjerne for en bedrift. Vårt utviklerteam, bestående av Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren (GSBOS), har inngått en avtale med IT-firmaet EVRY Norge AS (oppdragsgiver) om at vi skal utføre et prosjekt for dem.

EVRY Norge AS presenterte på NTNU i Gjøvik en oppgave der de ønsket å forbedre et eksisterende tilbud innenfor NaaS. Dette er en løsning hvor bedrifter og organisasjoner kan leie utstyr til å sette opp et nettverk som kunden selv eller EVRY Norge AS vedlikeholder. I denne pakken har EVRY Norge AS via samarbeidspartneren Aruba et overvåknings- og administrasjonsprogram for nettverk, kalt Aruba Central. Dette programmet henter inn store mengder data fra nettverket, håndterer oppsett og administrasjon av nettverk. Pakken settes opp med utstyr fra Aruba, og da oftest flere aksesspunkter. Antall og type er tilpasset basert på etterspørsel og radiokart fra området som skal dekkes av WiFi.

I dagens samfunn vil kunder som ønsker et slikt tilbud som oftest etablere ett eller flere trådløse nettverk. For eksempel vil et hotell gjerne ha et gjestenett og et ansattnett. Det blir da gjennom sine aksesspunkter at Aruba henter overvåkningsdata fra netteverket. Det er disse kundene EVRY Norge AS vil gi et bedre tilbud gjennom NaaS. Ved hjelp av Arubas REST-API kan man hente ut de ovennevnte dataene og prosessere disse etter EVRY Norge AS eller kundes SLA. EVRY Norge AS vil omtales som oppdragsgiver i resten av denne rapporten.

C.1.2 Prosjektmål

Resultatmål

- Oversiktsbilde over oppdragsgivers kunder, med tilhørende enheter og status på disse.
- Logging av nettverksinformasjonen for en gitt kunde
- Sortering etter alvorlighetsgrad med egendefinerte terskler (for eksempel en enhet er ute av drift, plutselig ingen tilkoblede klienter)
- Et sikkert system i henhold til kommende GDPR-bestemmelser og oppdragsgivers krav

Effektmål

GSBOS har ikke kompetanse til å estimere nøyaktig effekt på de utvalgte parametere vårt prosjekt vil ha. Det vil derfor bare bli uttrykt hvilke parametere som det estimeres forbedring på.

- Senke responstid ved feilmeldinger
- Øke tilgjengeligheten på statistikk fra WiFi-nettverket
- Kan lettere overholde SLA med den enkelte kunde

C.1.3 Rammer

Oppdragsgiver har ingen bestemte føringer på hvilke systemer som skal tas i bruk(per 22.01.18), bortsett fra Arubas teknologi og Aruba Central. Dette vil si at tjenesten vår er nødt til å kommunisere med Aruba Centrals API. Uten om dette skal løsningen være tilpasset større skjermer. Det er dermed opp til GSBOS å bestemme rammene for prosjektet, se seksjon C.2.3 for mer om dette.

Løsningen vil komme i et webapplikasjons-format og vil dermed måtte bli hostet på en server. Dette må GSBOS og oppdragsgiver diskutere. Dataflyt

vil bli diktert av dette valget. Mest sannsynlig vil vår løsning hente data rett fra serveren til Aruba i USA.

Prosjektavtalen ble signert tirsdag 09.01.18 og markerte starten av prosjektet for begge parter. Tidsrammen er fra dette tidspunktet til og med prosjektets slutt i mai. Oppdragsgiver forventer hittil ferdig produkt utlevert etter endt prosjektperiode. All kildekode laget for og under prosjektet er eiendeler av oppdragsgiver, dette står nærmere beskrevet i prosjektavtalen (vedlegg A).

C.1.4 Ressursbehov

Løsningen vil bli hostet på en server. Antagelig vil oppdragsgiver stille med dette. Videre trenger vi å holde en dialog med Aruba Central tech-support. Dette får vi tilgang til gjennom oppdragsgiver sin serviceavtale med Aruba.

C.2 OMFANG

C.2.1 Problemområde

I dag benytter oppdragsgiver seg av Aruba sitt overvåkning- og administrasjonssystem Aruba Central. Dette systemets webinterface krever mange museklikk for å få vist ønskelig informasjon. Oppdragsgiver synes at denne løsningen gir for lite informasjon på en tungvinn måte.

C.2.2 Oppgavebeskrivelse

Vår oppgave er å lage en oversiktlig webapplikasjon som krever ingen til få museklikk for å få vist informasjon over nettverkene som oppdragsgiver drifter. Løsningen skal gi et komplett bilde av nettverkenes status og omfang. Samtidig skal løsningen kunne generere rapporter som videre kan brukes i for eksempel problem management. Man skal kunne se dette i en storskjermvisning. Varsler skal genereres med for eksempel epost når et Access Point er nede. Andre lignende tilfeller som oppdragsgiver ønsker varsling på, vil bli varslet på lignende vis.

Løsningen vil hente data fra de forskjellige nettverkene som oppdragsgiver drifter gjennom Arubas API ved hjelp av HTTP-protokollens GET og POST metoder. Løsningen operer kontinuerlig og må derfor oppdatere sin informasjon ved jevne mellomrom.

C.2.3 Avgrensning

Vi har som mål internt i teamet å være ferdig med utviklingen av webapplikasjonen 27.04.18, på den måten har vi god tid til å ferdigstille rapporten med utgangspunkt i ferdig produkt. Mer om dette i seksjon C.6. I første møte med oppdragsgiver ble vi gjort oppmerksom på at det er problematisk å få kontakt med APIet, dette er en av de første og mest sentrale utfordringene vi må løse så fort som mulig, se figur C.3. GSBOS har blitt enige om å programmere i JavaScript for å lage webapplikasjonen, noe som kommer til å være en stor del av oppgaven vår. Ingen av medlemmene har tidligere erfaring med JavaScript som programmeringsspråk, og det vil derfor bli satt av tid i forkant av selve utviklingsperioden til å lære dette, samt finne passende rammeverk til prosjektet. Det er heller ingen i gruppen som har vert med på å produsere og publisere en webapplikasjon tidligere.
Oppdragsgiver har kommet med flere gode forslag til utvidet funksjonalitet, men dette blir ikke prioritert før hovedfunksjonaliteten er ferdig utviklet.

C.2.4 Problemstilling

Utvikle en webapplikasjon som ved færrest mulig museklikk viser mest mulig relevant data fra Aruba Central og som danner et oversiktlig bilde på storskjerm av disse dataene.

C.3 PROSJEKTORGANISERING

C.3.1 Ansvarsforhold og roller

Sammen i teamet ble vi enige om at Guro skal være teamleder. Hun er også medlem i utviklerteamet, og vi får en organisasjonsstruktur som vist på figur

C.1.

Bjørn Kråkevik representerer Evry Norge AS i denne sammenheng og er vår kontaktperson.
Tom Christensen jobber hos Evry Norge AS og har mye kunnskap om Aruba sine systemer. Han bistår oss i tekniske spørsmål rundt Aruba og Aruba Central.
Guro Storlien har vi internt i gruppen valgt som prosjektleder for utviklerteamet som består av Guro Storlien, Bjørn Ole Myrold, Snorre Rogne og Sondre Ahlgren.
Hovedkommunikasjonen mellom utviklerteamet og oppdragsgiver vil foregå mellom Bjørn Kråkevik og Guro Storlien. Diverse arbeidsoppgaver delegeres undervegs i prosjektet etter demokratisk avgjørelse internt i utviklerteamet. Det blir med andre ord ikke forhåndsbestemt hovedansvar for noen oppgaver med tanke på læringsutbytte.

C.3.2 Rutiner og regler i teamet

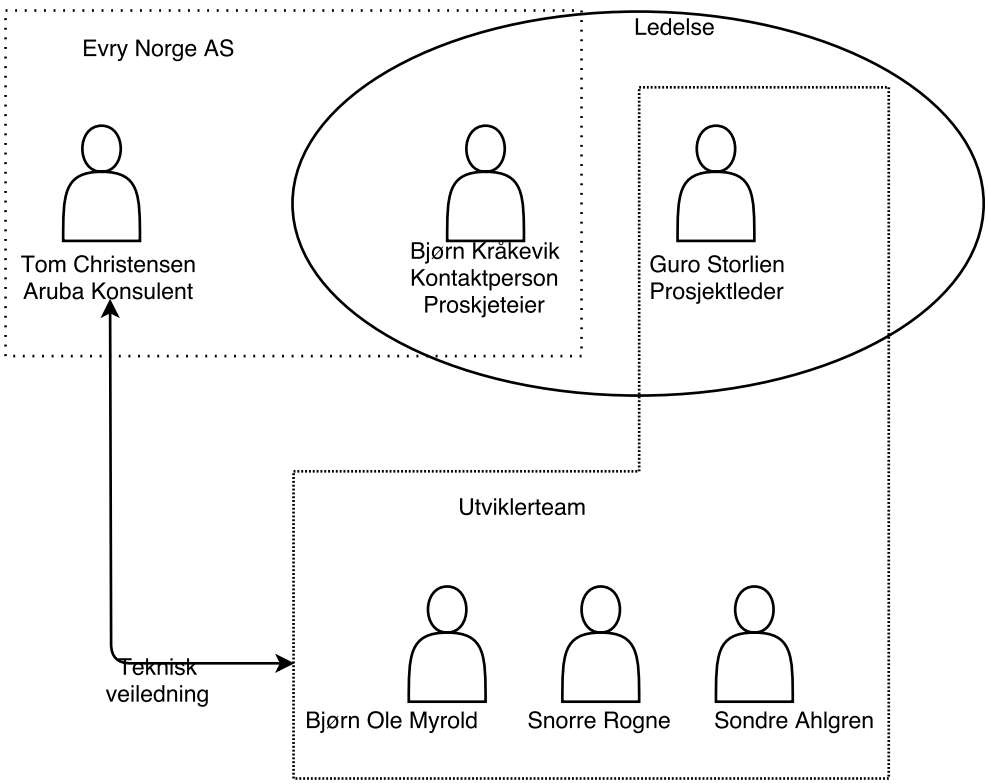
Grupperegler er definert og underskrevet som vedlegg D.

C.4 PLANLEGGING, OPPFØLGING OG RAPPORTERING

C.4.1 Valg av utviklingsmodell

Ved valg av utviklingsmodell er det flere argumenter for å velge en smidig utviklingsmodell.

- Oppdragsgiver er en pådriver for bruk av smidig utvikling.
- Utviklerene foretrekker smidig utvikling fremfor plandreven.
- Prosjektet har ingen definert sluttilstand, det vil alltid være mulig å utvikle noe mer.
- Utfordringene i prosjektet er vanskelig å tidsestimere, og vil derfor passe en smidig utvikling



Figur C.1: Organisasjonskart for prosjektet

- Prosjektet har fire utviklere i teamet, dette passer også en smidig utvikling.

Vi vurderer da modellene Kanban, eXtreme Programming (XP) og Scrum.

Kanban er en relativt enkel modell som baserer seg på en del relevante prinsipper. Open Kanban baserer seg på åpenhet rundt diskusjoner og avgjørelser, tildeling av ansvar ned til enkeltpersoner og god kommunikasjon innad i teamet. Dette er noe som er forenelig med vår teamstørrelse på fire og er prinsipper vi vil ta med oss i utviklingsarbeidet. Kanban Board er også et verktøy vi vil ta i bruk via utviklerverktøyet Taiga.io.

XP har også flere gode prinsipper som passer vårt utviklerteam på fire. On-site customer som vil si tett kontakt med oppdragsgiver vil være viktig i dette prosjektet. På tirsdager og torsdager har vi derfor tilgang på kontorplass hos oppdragsgiver. Dette vil gjøre at vi kan ha tett dialog med oppdragsgiver undervegs, men dette må også balanserens og gjøres via riktige kanaler. Modellen har også et prinsipp som heter parprogrammering, dette vil vi ta i bruk der vi ser det nødvendig hvor for eksempel kodens kvalitet er viktig, eller utviklingen er spesielt krevende. XP tar også for seg en balansert arbeidshverdag hvor skippertak og mye overtid skal sjeldent forekomme. Dette er også prinsipper som teamet vil bruke og som er viktig for en god gjennomføring av bacheloroppgaven.

Scrum tilbyr en stor plattform og mange verktøy til smidig utvikling. Modellen beskriver en gitt struktur med sprinter som passer oppgavens art. Oppdragsgiver har få punkter i kravspesifikasjonen. Løsningen skal kort sagt gi et enkelt og oversiktlig grenseutsnitt GUI som gir relevant informasjon med tanke på drift og overvåkning av WiFi-tjenesten. Iterativ utviklingsmodell passer godt til dette prosjektet hvor det er nødvendig å legge til nye arbeidsoppgaver underveis i prosjektet. Teamet ønsker også en utviklingsmodell basert på scrum da dette passer oss i form av arbeidsmetodikk, selvstendighet og beslutningsdyktighet. Scrum vil derfor være utviklingsmoddellen vi i hovedsak vil følge hvor prinsipper fra kanban og XP vil bli integrert i Scrum-modellen.

Utfordringen med Scrum vil være at teamet ikke har brukt smidig utvikling i en lignende oppgave før, ei heller erfaring med denne typen utviklingsprosjekt. På grunn av dette vil det å estimere de ulike Product Backlog Item's i forkant av hver sprint være en utfordrende oppgave. Vi har derfor valgt å estimere arbeidsmengde med verdier fra 1-5 i Taiga.

Rollefordeling:

- Product Owner - Evry Norge AS ved Bjørn Kråkevik
- Scrum Master - Guro Storlien, Prosjektleder
- Scrum Team - Storlien, Alghren, Myrold og Rogne

C.4.2 Hovedinndeling av prosjektet

Oppstarten vil bære preg av å bli kjent med Aruba og dagens løsning. Videre må teamet bli kjent med verktøyene og programmeringsspråkene vi skal bruke. Det er viktig at vi raskt får kontakt med API settet til Aruba og at vi klarer å hente ut den informasjonen vi ønsker. Informasjonen vi har hentet ut skal videre behandles og legges inn i en webapplikasjon. Underveis skal det meste dokumenteres og redgjøres for, og hele prosjektet vil ende i en sluttrapport som utformes til dels under hele prosjektperioden. Undervegs vil vi jobbe med database, servere, innlogging, testing og sikkerhet for å lage så helhetlig produkt som mulig.

Plan for statusmøter

Ved avslutningen på hver Sprint vil vi ha retrospektive møter. Tre av disse vil være utvidet til statusmøte med rapport. Møtene har størst viktighet i utviklingsfasen og vil derfor forekomme hyppigere i denne perioden. Datoene for gjennomføringen av statusmøter blir:

- Sprint 2: Onsdag 21. februar
- Sprint 4: Fredag 23. mars
- Sprint 6: Fredag 27. april

Beslutninger

Det er viktig at utviklerteamet tilstreber å ta gode og korrekte beslutninger selvstendig eller innad i teamet. Ved usikkerhet skal spørsmålet drøftes i teamet, eventuelt med veileder eller oppdragsgiver.

C.5 KVALITETSSIKRING

C.5.1 Standard/verktøy-bruk

- LaTeX brukes hovedsaklig som skriveverktøy for rapportskriving, møtereferat og andre dokumenter med tilknytning til arbeidsprosessen.
- Taiga er valgt som utviklingsverktøy siden det passer bra til smidig utvikling
- Toggl benyttes til å føre timelister for prosjektdeltakerne
- Visio er standard for utarbeiding av diverse grafer og diagrammer. I enkelte tilfeller hvor det forekommer som gunstig og/eller hensiktsmessig, kan draw.io benyttes
- Et git-repository vil brukes som versjonkontroll gjennom hele utviklingsfasen. Vi har her bestemt oss for å bruke bitbucket, siden vi som utviklere får tilgang til student-lisens gjennom NTNU
- Store deler av utvikling vil foregå i Javascript som programmeringsspråk, innslag av HTML og CSS
- Vi velger å hoste vår løsning ved hjelp av Node.js

C.5.2 Dokumentasjonskrav

Arbeidsprosess

Store deler av selve arbeidsprosessen som teamet foretar seg skal dokumenteres, slik at det blir enklere å se tilbake på ting som er blitt besluttet og utført gjennom en gitt sprint. Toggl vil bli benyttet av teamet for å føre timelister med intuitive beskrivelser om hva som er blitt gjort i aktuelle tidsrom. I loggboken føres møtereferater og dagslogger.

Ferdigstilt produkt

Vi oppretter en wiki/brukermanual i tillegg til allerede eksisterende API-doc for det ferdige produktet slik at oppdragsgiver skal få grunnleggende forståelse tidlig etter at produktet først tas i bruk.

Kildekode

All kildekode skal dokumenteres etter standard for det aktuelle programmeringsspråket som tas i bruk under utviklingsprosessen. For språk som har integrerte dokumentasjonsprogrammer skal dette alltid benyttes. Øvrig gjelder god struktur og sammenheng mellom kode og kommentarer etter en god standard som teamet har blitt enige om på forhånd.

Møter og veiledning

Prosjektleder påtar seg rollen som sekretær under møter med både oppdragsgiver og veileder. Øvrige teammedlemmer oppfordres likevel sterkt til å ta egne notater. Etter et møte skal alle på teamet gå sammen og renskrive møtenotatene til et forståelig og ferdigstilt referat.

Utviklingsperioden

Det vil bli avholdt daily scrum møter på ca. ti minutt for å holde alle på teamet oppdatert på den helhetlige prosessen.
Etter hver sprint vil det også holdt et retrospective møte for alle utviklere, eller et review møte med eventuelle personell hos oppdragsgiver som har deltatt aktivt med veiledning. Disse møtene skal også dokumenteres.

C.5.3 Testing

Vi kommer til å benytte oss av unit-testing ved hjelp av Jest for erfaringens del, men vi binder oss ikke til å bruke dette fast for hele applikasjonen, da vi ikke er kjent med omfanget det måtte innebære å skrive tester for alle modulene i prosjektet.

C.5.4 Risikoanalyse

Identifikasjon og analyse

All risiko som kan gjøres tiltak mot fra begynnelsen av markeres med en «*» i nummereringen. Vi benytter følgende standard for vår risikoanalyse, hvor hver grad har fått et tall som brukes i risikoberegningen:

- **Sannsynlighet:** Særdeles liten 1, Liten 2, Trolig 3, Stor 4, Særdeles stor 5
- **Konsekvens:** Trivielt 1, Mindre alvorlig 2, Kritisk 3, Meget alvorlig 4, Katastrofalt 5

Produktet av disse gir så en indikator på risikograden til de ulike scenario. Disse tolkes på følgende måte:

- Lav risiko/grønt nivå: (1-4)
- Middels risiko/gult nivå: (5-12)
- Høy risiko/rødt nivå: (13-25)

Se figur C.2 for full oversikt over identifiserte risikoer

Planlagte tiltak

Figur C.3 viser en oversikt over de identifiserte prosjektrisikoene sortert etter risikograd, med tilhørende planlagte tiltak for disse.

C.6 PLAN FOR GJENNOMFØRING

I utviklingsfasen av prosjektet planlegger vi å jobbe med seks sprinter med et intervall på 10 dager hver. Det vi bli lagt inn et unntak for dette på grunn av planlagt påskeferie for utviklerteamet: Sprint 4 avsluttes fredag 23.03.18 (uke 12), og sprint 5 vil da starte på igjen tirsdag 03.04.18 (uke 14). Dette medfører at sprint 5 vil effektivt bli 9 arbeidsdager lang. Etter endt påskeferie vil hovedfokuset gå over fra utvikling til skriving av prosjektrapporten, og ved

#	Scenario	Sannsynlighet	Konsekvens	Risiko
1*	Sykdom innad i prosjektgruppen som fører til fravær over korte tidsperioder	Stor	Mindre Alvorlig	Middels - 8
2	Sykdom/skade/andre årsaker fører til at et gruppemedlem må forlate prosjektet	Særdeles liten	Katastrofal	Middels - 5
3*	Kildekode går tapt grunnet systemkræsj, uaktsomhet eller liknende grunner	Liten	Katastrofal	Middels - 10
4*	Svak dokumentasjon	Trolig	Kritisk	Middels - 9
5*	Prosjektet blir ikke fullført innen gitte tidsrammer	Trolig	Meget Alvorlig	Middels - 12
6	Uenighet innad i gruppen angående teknisk utførelse av enkelte arbeidsoppgaver	Trolig	Trivielt	Lav - 3
7	Prosjektgruppen får ikke etablert forbindelse med Arubas API innen rimelig tid	Stor	Meget Alvorlig	Høy - 16

Figur C.2: Identifiserte risikoer

Scenario	Tiltak
Prosjektgruppen får ikke etablert forbindelse med Arubas API innen rimelig tid	Utforske muligheter til å utføre andre arbeidsoppgaver som en midlertidig løsning. Bruke tiden godt til å lære seg opp i relevante emner.
Prosjektet blir ikke fullført innen gitte tidsrammer	Sørge for god planlegging og gjennomføring av dette. Sørg for å ikke bite over mer enn vi kan tygge
Kildekode går tapt grunnet systemkræsj, uaktsomhet eller liknende grunner	Holde på gode backup rutiner
Svak dokumentasjon	Regelmessig minne hverandre på å dokumentere hva som blir gjort
Sykdom innad i prosjektgruppen som fører til fravær over korte tidsperioder	Gruppen tar godt vare på hverandre, og sørger for at alle får tilstrekkelig med søvn og næring
Sykdom/skade/andre årsaker fører til at et gruppemedlem må forlate prosjektet	Gå inn i samtale med veileder og oppragsgiver for å se an muligheten for å redusere omfanget på prosjektet noe.
Uenighet innad i gruppen angående teknisk utførelse av enkelte arbeidsoppgaver	Demokrati. Ved likt antall stemmer har prosjektleder rett til å foreta en avgjørende beslutning

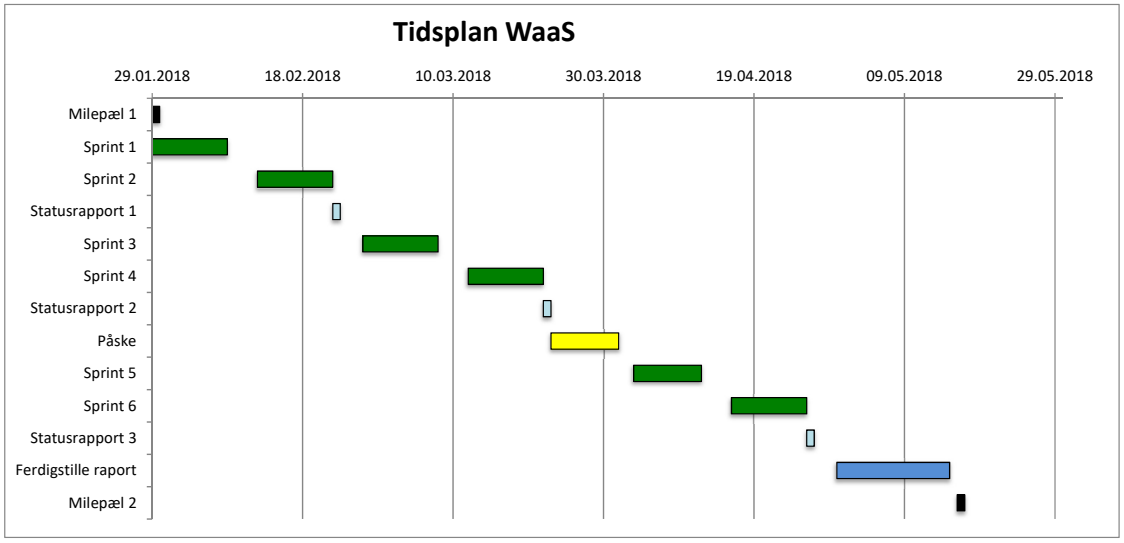
Figur C.3: Forslag til tiltak

slutten av uke 17 (fullført sprint 6) planlegger vi å være helt ferdig med vårt produkt, noe som gir oss litt i overkant av to uker til å ferdigstille rapporten for innlevering 16.05.18.

C.6.1 Gantt-skjema

Oppgave	Start	Slutt	Varighet (dager)
Milepæl 1	29.01.2018	30.01.2018	1
Sprint 1	29.01.2018	09.02.2018	10
Sprint 2	12.02.2018	23.02.2018	10
Statusrapport 1	22.02.2018	23.02.2018	1
Sprint 3	26.02.2018	09.03.2018	10
Sprint 4	12.03.2018	23.03.2018	10
Statusrapport 2	22.03.2018	23.03.2018	1
Påske	23.03.2018	02.04.2018	9
Sprint 5	03.04.2018	13.04.2018	9
Sprint 6	16.04.2018	27.04.2018	10
Statusrapport 3	26.04.2018	27.04.2018	1
Ferdigstille raport	30.04.2018	16.05.2018	15
Milepæl 2	16.05.2018	17.05.2018	1

Figur C.4: Datooversikt for tidsplanen i figur C.5



Figur C.5: Tidsplan

D Grupperegler

Grupperegler

Guro Storlien, Bjørn Ole Myrold, Sondre Ahlgren, Snorre Rogne

11. January 2018

1 Introduksjon

Grupperegler for utføring av Bachelor prosjekt, med tilhørende utvikling og rapportskriving.

2 Arbeidskrav

Det forventes minimum 25 arbeidstimer per pers i uken under hele prosjektperioden. Det skal allikevel være av høy prioritet å fullføre tidspressende arbeidsoppgaver som burde bli gjort. Helg kan til nød tas i bruk. Ved planlagte arbeidstimer, er gruppen pålagt å møte i tidsrommet, ellers kan man jobbe til alle døgnets timer. Hovedsaklig vil arbeidstimer være hverdager 08:00 til 16:00. Arbeid utover dette kan ikke forventes av alle (jfr. noen har familiære/jobbmessige/andre plikter). Dersom det skulle skje at vi havner langt bak med arbeidsoppgaver, kan gruppa i fellesskap bli enige om å jobbe overtid (for eksempel 10 timer i helga).

Gruppen har møteplikt på prosjekt-/statusmøter hver uke. Er et gruppemedlem utilgjengelig eller lignende, må medlemmet gi beskjed i god tid slik at møtet evt. kan flyttes. Dato for møter settes fortløpende. Ved enighet om dato og tid, er møtet obligatorisk. Vi tilstreber møtetider hvor alle gruppemedlemmer er tilgjengelig. Det forventes aktiv deltagelse i arbeidet fra samtlige medlemmer. Gruppen har et felles mål om å nå frister (gruppen ønsker gode marginer, slik at utfordringer kan overkommes).

Ved føring av referat på statusmøter skal disse inneholde:

- Fattede avgjørelser
- Hva som er utført/ferdig
- Fremdriftsplan mot neste møte
- Ting som må gjøres oppdateres og frister settes m.m.

Standardformat på dokumenter:

- PDF
- LaTeX(Sharelatex) brukes som verktøy
- Fontstørrelse: 11

3 Uforutsette problemer

Å være forberedt/gjort oppgaver til avtalt tid gi tidlig beskjed om man "henger etter", og evt. enighet om at "dette er greit"/kan hentes inn. Ved faglige uenigheter skal gruppen forsøke å komme til enighet, evt. fattes en flertall-savgjørelse. Siden vi er en gruppe på 4 og avgjørelser kan ende med en 2 mot 2 situasjon, har gruppeleder fullmakt til å fatte en tidlig beslutning etter å ha hørt argumentasjon fra begge sider. Gruppemedlemmene må tidlig si ifra om de ikke er fornøyd med innsatsen til enkelte gruppemedlemmer. Dersom det skulle oppstå et større problem: Samtale(r) mellom alle gruppemedlemmer om problemet, der regelbruddet blir påpekt. Konkret (tidsfrister, forventet arbeid(sinnsats), m.m.) om hva som kreves for å "bøte på skaden". Sette inn tiltak mot regelbruddet. Det er ikke lov å ekskludere et gruppemedlem.

Ved sykdom eller fravær under 7 dager må denne personen ta igjen tapt arbeid påfallende uke. Dersom fraværet går over 7 dager, skal veileder kobles inn i beslutninger og dialog med instituttet vil avgjøre om personen skal gjennomføre bacheloroppgaven.

4 Signaturer:

Guro Storlien

Sondre Ahlgren

Bjørn Ole Myrold

Snorre Rogne

E Utvidede kodelistinger

E.1 update_device

```
1  CREATE OR REPLACE update_device (
2      p_cus_id IN NUMBER,      -- Aruba customer id
3      p_type VARCHAR2          -- type of device (ap/switch)
4    ) IS
5
6    -- vars for retrival of data
7    v_jsonvalue          apex_json.t_values;
8    v_respvalue          CLOB;
9
10   -- cryptovars
11   v_at_raw             RAW;
12   v_at_key             RAW;
13   v_at                 VARCHAR2;
14
15   -- general vars
16   v_base_url           VARCHAR2;
17   v_req_type           VARCHAR2;
18   v_table_cus_id       NUMBER;
19
20   -- vars for each field in the DB
21   v_serialNum          VARCHAR2(200);
22   v_macAdress          VARCHAR2(200);
23   v_ipAdress           VARCHAR2(200);
24   v_model              VARCHAR2(200);
25   v_name               VARCHAR2(200);
26   v_firmwareVersion    VARCHAR2(200);
27   v_lastModified_temp  NUMBER;
28   v_lastModified       DATE;
29   v_status             VARCHAR2(20);
30   v_downReason         VARCHAR2(200) := NULL;
31
```

```
32
33  BEGIN
34    -- retrive sysparam
35    SELECT AWS_SYP_ARUBA_BASE_URL
36    INTO v_base_url
37    FROM AWS_SYSTEM_PARAMETER
38
39    -- get raw at and primary key for given customer
40    SELECT AWS_CUS_AT, AWS_CUS_ID
41    INTO v_at_raw, v_table_cus_id
42    INNER JOIN AWS_CUSTOMER_DETAIL ON (AWS_CUS_CUD_ID = AWS_CUD_ID)
43    FROM AWS_CUSTOMER
44    WHERE AWS_CUD_CUSTOMER_ID = p_cus_id;
45
46    -- get at key
47    v_at_key := get_customer_key('access_token', p_cus_id);
48
49    -- decrypt
50    v_at := decrypt_varchar(v_at_raw, v_at_key);
51
52    -- The requeststring is different from type
53    IF p_type = 'ap' THEN
54      v_req_type = 'aps';
55    END;
56    IF p_type = 'switch' THEN
57      v_req_type = 'switches';
58    END;
59
60    -- ask API for devices of the given type
61    -- EX: https://app1-apigw.central.arubanetworks.com/
62    --     monitoring/v1/aps?access_token=<some at>;
63    v_respvalue := get_http_response(
64                         v_base_url
65                      || 'monitoring/v1/'
66                      || v_req_type
67                      || '?access_token='
68                      || v_at
```

```
69                  );
70
71    apex_json.parse(
72      p_values => v_jsonvalue,
73      p_source => v_respvalue
74    );
75
76    FOR v_device in apex_json.get_members(
77                      p_path => v_req_type,
78                      p_values => v_jsonvalue
79                    );
80
81    -- update the table with all devices of
82    -- the given type
83    LOOP
84      -- retrive data from JSON-object
85      v_serialNum         := apex_json.get_varchar2(
86                               p_values => v_device,
87                               p_path => 'serial'
88                             );
89      v_macAdress         := apex_json.get_varchar2(
90                               p_values => v_device,
91                               p_path => 'macaddr'
92                             );
93      v_model             := apex_json.get_varchar2(
94                               p_values => v_device,
95                               p_path => 'model'
96                             );
97      v_ipAdress          := apex_json.get_varchar2(
98                               p_values => v_device,
99                               p_path => 'ip_address'
100                            );
101     v_name              := apex_json.get_varchar2(
102                              p_values => v_device,
103                              p_path => 'name'
104                            );
105     v_firmwareVersion   := apex_json.get_varchar2(
```

```
                              p_values => v_device,
                              p_path => 'firmware_version'
                            );
      v_lastModified_temp := apex_json.get_varchar2(
                              p_values => v_device,
                              p_path => 'last_modified'
                            );
      -- Converts retrieved NUMBER-formated VARCHAR2-value to DATE.
      v_lastModified := epoch_to_date(TO_NUMBER(v_lastModified_temp));

      v_status            := apex_json.get_varchar2(
                              p_values => v_device,
                              p_path => 'status'
                            );
      -- should stay 'NULL' if status = 'up'
      IF v_status = 'down' THEN
      v_downReason        := apex_json.get_varchar2(
                              p_values => v_device,
                              p_path => 'down_reason'
                            );
      END IF;

      -- insert or update DB with data
      BEGIN
        LOOP
          BEGIN
            MERGE INTO AWS_DEVICE USING dual
            ON ("AWS_DEV_SERIAL" = v_serialNum)

            WHEN MATCHED THEN UPDATE SET (
              "AWS_DEV_DEVICE_TYPE"       = p_type,
              "AWS_DEV_MACADDR"           = v_macAdress,
              "AWS_DEV_IP_ADDRESS"        = v_ipAdress,
              "AWS_DEV_MODEL"             = v_model,
              "AWS_DEV_NAME"              = v_name,
              "AWS_DEV_FIRMWARE_VERSION"  = v_firmwareVersion,
              "AWS_DEV_LAST_MODIFIED"     = v_lastModified,
```

```
143             "AWS_DEV_STATUS"            = v_status,
144             "AWS_DEV_DOWN_REASON"       = v_downReason
145           )
146
147           WHEN NOT MATCHED THEN INSERT (
148             "AWS_DEV_CUS_ID",
149             "AWS_DEV_SERIAL",
150             "AWS_DEV_DEVICE_TYPE",
151             "AWS_DEV_MACADDR",
152             "AWS_DEV_IP_ADDRESS",
153             "AWS_DEV_MODEL",
154             "AWS_DEV_NAME",
155             "AWS_DEV_FIRMWARE_VERSION",
156             "AWS_DEV_LAST_MODIFIED",
157             "AWS_DEV_STATUS",
158             "AWS_DEV_DOWN_REASON"
159           )
160           VALUES (
161             v_table_cus_id,
162             v_serialNum,
163             p_type,
164             v_macAdress,
165             v_ipAdress,
166             v_model,
167             v_name,
168             v_firmwareVersion,
169             v_lastModified,
170             v_uptime,
171             v_status,
172             v_downReason
173           );
174
175           EXIT; -- success? -> exit loop
176       EXCEPTION
177         -- the entry was concurrently deleted
178         WHEN NO_DATA_FOUND THEN
179           -- exception? -> no op, i.e. continue looping
```

```
180             NULL;
181
182           -- an entry was concurrently inserted
183           WHEN DUP_VAL_ON_INDEX THEN
184             -- exception? -> no op, i.e. continue looping
185             NULL;
186       END;
187
188     END LOOP;
189 END update_device
```

F Brukermanual

Brukermanual for NaaS-overvåkingsverktøy

Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren

16. mai 2018

F.1 Introduksjon

Dette dokumentet vil gå gjennom funksjonalitet i NaaS-overvåkingsverktøy laget av Guro Storlien, Snorre Rogne, Bjørn Ole Myrold og Sondre Ahlgren under et bachelor prosjekt for EVRY Norge AS.

F.2 Legge til ny kunde

I menyen, velg **Edit Customer**. Oppe til høyre på siden: **Create**. Du får da opp et skjema som skal fylles ut:

- **Customer Id**: Aruba sin egen **customer id**
- **Client Id og Client Secret**: Brukes til autentisering av denne kunden til API-et. For å generere disse må du gå inn på Aruba Central. Gå inn på kunden det gjelder. Gå til **Maintainance - API Gateway - Authorized Apps & Tokens**. Her kan du legge til en «app» som genererer client id og client secret.
- **Customer Relation**: Brukerdefinert beskrivelse av forholdet med denne kunden. Denne teksten vil bli vist ved siden av kunden i oversikten.
- **AP down param**: På en skala fra 1 - 3 hvor kritisk er det at ett aksesspunkt er ned på denne kunden.
- **Switch down param**: På en skala fra 1 - 3 hvor kritisk er det at en switch er nede på denne kunden.
- **Thrustworthy param**: Desverre ikke implementert enda. Dette tallet påvirker ikke prioriteringen slik som det står nå.

F.3 Legge til ny APEX bruker

Du må være innlogget som admin(i STUDENT workspace) for å kunne legge til en ny APEX bruker.

Når du er på hovedsiden: Oppe til høyre er det en meny med en siluett med en skiftnøkkel. I den menyen velger du **Manage Users and Groups**. Der får du en oversikt over alle brukerne på denne workspace. Oppe til høyre er det en blå knapp **Create User**. Her fyller du ut nødvendige credentials. Under «Account Privileges» setter du «Team Development Access» til **no** og lar resten stå som det er, for å gi denne brukeren reduserte privilegier.

F.4 Report

En **Report** er en samling av modifiseringer du gjør på tabellen, noen eksempler under. Report lages ved å trykke på **Actions** - **Report** - **Save As**. Tre kategorier av report finnes: Default ..., Alternative vises til andre brukere, Private vises bare til den brukeren den blir laget på. Viktig: Når du har valgt hvilke highligths og sorteringer du skal ha så må du gå inn på **Report** og trykke **Save**. Det samme gjelder for sletting, man må stå i den **Report** det gjelder.
Fungerer ikke highlighting etter endring? Refresh siden.

Flytte på en kolonne

Flytt musepekeren helt til venstre på navnet til kolonnen. Musepekeren skal bli til en neve som griper tak(ikke en neve med pekefingeren ut), trykk og dra til ønsket lokasjon.

Sortering

Hver rad kan sorteres på forskjellig verdi av en spesifikk kolonne. Høyre på en kolonnes navn så er det en pil opp og en pil ned for respektivt ascending/descending.

Highlighting

Fargelegg rad(er) som har en spesifikk verdi på en kolonne. **Actions** - **Format** - **Highlight**. Vinduet som kommer opp har oversikt over alle lagrede

highlights på venstresiden. Ny highlight lages med pluss tegn nede til venstre.

F.5 Eksportere networks eller devices til .csv/.html

Venstre kolonne på forsiden, under drop down menyen **Overview** finnes oversikt for: Networks og Devices. På toppen av tabellen er det en drop down meny **Actions**. Velg **Download**.

F.6 Feilsøking

Deadlock forekommer når flere prosesser prøver å aksessere samme ressurs. Dette løser seg selv ved neste syklus av scheduled joben som forårsaket feilen.

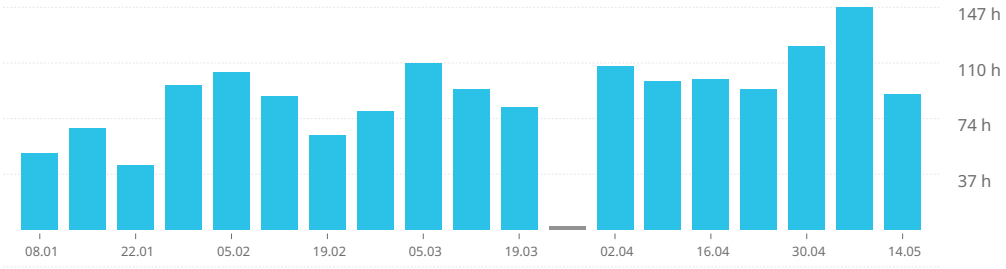
Gjenerelle feil kan forekomme. Sjekk log tabellen sin error message. Det kan hende at feilmeldingen fra API-et ligger der og gir informasjon om hva som gikk galt. Ved feil akkreditiv ve å legge inn kunde vil meldingen være «couldn't find customer».

G Timelogg

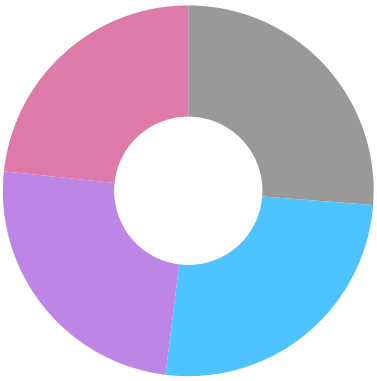
Summary report

2018-01-09 - 2018-05-15

Total 1616 h 46 min

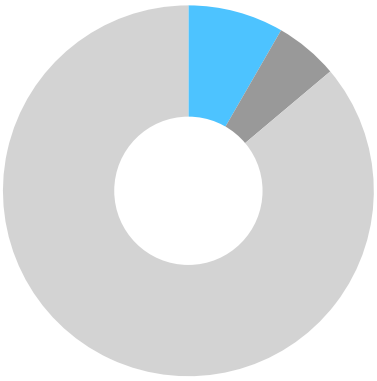


Users



- Snorre 424:00:46
- Storlien Guro 416:06:33
- Sondrah 399:28:51
- Bjørn Ole Myrold 377:10:00

Time entries



- Rapportskriving 134:45:01
- GS - rapport 89:30:00
- Other 1392:31:09

Users / Time entries	Duration
Storlien Guro	**416:06:33**
GS - Web Rep update	2:30:00
GS - web rep + møte	3:30:00
GS - Web/REP GUI SQL	5:30:00
GS - Web/REP, debugge GUI SQL	8:00:00
GS - Web GUI Doc	7:00:00
GS - Verktøy + S1	5:00:00
GS - Veiledning 2	0:45:00
GS - Veiledning 1	0:30:51
GS- Veiledning	0:45:00
GS - Update, Trigger info	23:14:00
GS - Update funksjoner	7:11:00
GS - Update correct info + GUI	8:00:00
GS - update and refrech	3:00:00
GS - Update all, token problem + veil	5:00:00
GS - Teknisk møte m Evry	3:43:00
GS - Statusrapport + mail	1:30:00
GS - SQL TABELS API	7:00:00
GS - SQL Package	1:30:00
GS - SQL GUI + Veil	2:00:00
GS - SQL Dev. Package	7:00:00
GS - SPM Taiga	1:00:00
GS - SPM, Retro + status	1:30:00
GS - Retro /SPM	3:00:00
GS - Rapport token mail	2:00:00
GS - rapport	89:30:00
GS - Prosjektplan + S1	4:43:00
GS - Prosjektplan Presantasjon + møter	4:00:00
GS - Prosjektplan Presantasjon	11:06:00
GS - Prosjektplan	9:24:00
GS - Postman Token Refrech ok	2:44:00
GS - Postman AP React	3:12:00
GS - Postman AP info	3:20:00
GS - Oppstartsmøte Granvold	4:55:00
GS - Opdate funk	5:00:00
GS- Møte med oppdragsgiver	1:00:00
GS - Modelering	4:00:00
GS - Lynkurs 2	1:00:00

GS - LYNKURS	1:30:00
GS - logging and GUI	9:00:00
GS - log err	8:00:00
GS - Lese Aruba C	2:00:00
GS - JS+ Redux + S1	7:53:00
GS - JS	3:46:00
GS - I18N + veil	3:00:00
GS - I18N + GUI	7:00:00
GS - GUI CUSTOMER DETAIL	14:00:00
GS - Funksjoner og login moddell	2:45:00
GS - Forarbeid og prosjektplan	8:14:45
GS - forarbeid	2:00:00
GS - DS, SR, SPM + Skrivekveld	10:00:00
GS - debugge log	1:00:00
GS - Auth error og Sheggerdudle	8:00:00
GS - Aruba oppsett	3:00:00
GS - AP Mail React	3:24:00
GS - API, React Rapport	5:10:00
GS - API og URL	1:00:00
GS - API Auth og URL	7:00:00
GS - APEX SQL DB-mod	5:00:00
GS - APEX, Rapport, React JS	4:52:57
GS - APEX, Rapport, Mail, VP	4:30:00
GS - APEX DB-mod Møte	8:00:00
GS - APEX DB	13:00:00
GS - APEX, CLOB, Packages og JSON	6:38:00
GS - APEX API Møte	4:20:00
GS - APEX	8:00:00
Sondrah	**399:28:51**
Vitenskapelig tilnærming til problemstilling	2:00:00
veiledningsmøte + rapport + fikse jobs	3:00:00
veiledning	0:45:00
Veilding	0:25:00
updateNetwork kall i aiur-trigger	2:30:00
updateNetwork + clarity	7:55:00
update_bandwidth_networks	3:30:00
update_bandwidth + diverse	8:15:00
trg_customer_detail_aiur ferdig	5:50:00
testet rundt feil ved autentisering + job for refresh msp token	2:00:00

Statusrapport + http header	4:20:00
Skrive Omfang i prosjektplan + litt forstå API	3:39:00
Skrev innledning + planlegging av neste uke	3:50:00
Sette seg inn i HTTP & React	2:00:00
Satt opp DB(delvis) + full kontakt med API	8:00:00
Satt meg inn i Postmann og autentisering	3:15:00
Rette opp i prosjektplan etter veiledning	3:00:16
Retrospective, veiledning og sprint planning	3:30:00
retrospective + sprint 4 planning	4:05:00
retrospective + rapport	1:30:00
retrospective meeting + planlegging av løpet videre	6:45:00
renskrive kode + diverse	8:00:00
refresh_token ferdig + 1. utkast innledning og kravspec	5:15:00
Reasearche APEX + forberede engelsk presentasjon	4:00:00
React + Node.js	6:00:00
rapport + unit testing	7:20:00
Rapportskriving	10:45:01
rapport + refresh_tokens	6:00:00
rapportgjennomgang	8:00:00
Prosjektplanlegging	3:03:23
Prosjektplan + byggd backlog	2:49:19
prosess	8:00:00
Postmann	4:00:00
Plan for gjennomføring + argumentere for Scrum	3:30:48
Oppsumering + retrospective + planlegging	3:45:00
Oppstartsmøte	5:00:00
metode	3:00:00
Lynkurs	1:00:00
litt mer kartlegging + fått inn info i DB	8:00:00
Litt APEX + presentasjon	3:00:00
Litt APEX	1:00:00
Lese tidligere bachleoroppgave + kartlegge omfang	1:04:39
Lese tidligere bachleoroppgave	0:38:22
Lage oversikt over API url-er + Nawaar viste APEX	3:20:00
Lage grupperegler	2:00:00
Lære seg Nodejs	2:00:00
Lære seg JacaScript	2:42:58
Lære javascript + diverse skriving + fikse latex problem	3:09:00
kartlegge struktur på applikasjonen	2:15:00

jobs up and running + rapport + veiledningsmøte	4:00:00
insert og update row	7:15:00
Innføring oracle DB + oppklarte Aruba problem	8:00:00
Innføring i hva Evry har gjort av forarbeid	1:00:00
Innføring APEX + oppsett av rapport	4:00:00
Google løsning på certificate problem	3:25:00
gjennomgått tilbakemeldinger	4:00:00
Gantt skjema + rapportskriving	2:00:30
Fortsett autentisering + henter data fra API i package	7:50:00
Forstå OAuth2	1:30:00
Forstå API	5:20:00
Fornyet access token	3:20:00
Finpuss av prosjektplan	3:00:00
finpuss	11:00:00
fikset feil i aiur trigger + refresh msp token procedure	8:25:00
diverse	3:00:00
Diskusjon og ferdeling av forprosjektet	3:00:00
Design + diskutere oppsett	2:28:51
Demo om Aruba Networks	4:00:00
debugging	7:45:00
Datamodelering + PL/SQL	3:25:00
Database moddelering + kravspec	8:00:00
database + arbeidsprosess	9:00:00
customer_detail trigger + network_update	8:00:00
customer_detail trigger	2:10:00
codeweekend	10:00:00
brukermanual + exception	8:00:00
Bestemmelse av diverse rammeverk	1:10:44
bandwidth + rapport	6:50:00
avslutning kappitell	7:15:00
avslutning	7:15:00
Autentisering ferdig	7:45:00
Autentisere fra package	3:21:00
arbeidsprosess	7:00:00
apex + rapport	3:30:00
APEX	5:45:00
AIUR ferdig. Fikset hierarkisk oppdatering	8:00:00
Snorre	**424:00:46**
Veiledningsmøte	0:39:36

Veiledning	1:00:40
veil 1	0:25:33
Utforske Arbua central	1:26:39
update funk, sprint retro	7:05:51
UPDATE-funksjoner	8:10:25
update-funk	3:50:42
UPDATE_CUSTOMER	4:13:42
Testet ulike requests i swagger og postman	5:32:23
Teknisk opplæring i Aruba Central hos Evry	4:01:53
Studerer tidligere bacheloroppgave	0:38:34
Sprint retrospective, Sprint planning	3:45:38
sprint planing, sprint retro, rapport - impl.	7:45:05
Sprint 2 retrospective, planning meeting	4:49:26
Skrivekveld	4:20:08
Skrevet på rapport, planlagt neste uke	3:45:41
Retrospektive, statusrapport, korrektur	3:41:53
Rapport, ymse koding	7:04:19
Rapport, ymse koding	7:17:32
Rapport, veiledning	4:52:27
Rapportstruktur og korrektur, avslutningskapittel	10:49:21
rapportskriving	7:56:54
rapport, sekvensdiagram	10:48:00
rapport-review, refaktorisering, veiledning	4:29:47
Rapport, renskriving av møtereferat	7:56:49
Rapport	14:38:42
Prøvd å forstå mer av logikken til Aruba Central. APEX	7:50:04
Prosjektplan review, utbedringer	0:58:12
Prosjektplan - Kvalitetssikring forts.	2:37:44
Prosjektplan - Kvalitetssikring	2:51:53
Prosjektplan gjennomgang	1:41:23
Prosjektplan - Finpuss, avklare dokumentstruktur	4:02:44
Prosjektplan	3:09:29
Postman, testet ulike get/post requests mot API	4:33:34
Postman	3:19:31
Plan for gjennomføring	2:37:29
Oppstartsmøte med gruppemedlemmer	2:29:57
Oppstartsmøte med Evry	1:59:42
Opplæring, React/JS	2:31:20
Opplæring, JS	5:02:45

Opplæring, javascript	3:52:27
Opplæring i Apex's info.flyt, databaser og generell oppbygning	8:00:14
Møte - Retrospective, veiledning, sprint planning	3:30:00
Møte med ansatt hos Evry	1:05:45
Metode, spesifikasjon, veiledning	4:51:35
Lynkurs	0:44:28
Lest opp på hvordan foreta API-kall fra PL/SQL. Prøvd diverse script	2:32:27
Lest opp på dokumentasjon, blitt kjent med andre sitt arbeid	3:55:01
Lest opp på APEX, laget presentasjon om prosj.plan	4:09:00
Lage create-trigger som oppdaterer og inserter informasjon i customer-tabeller	7:58:34
Krav til løsningen	7:20:37
Kravspec + design	8:14:40
Korrektur, sammendrag	9:49:04
Korrektur	7:53:46
Konstruert diagrammer, korrektur, kvalitetssikring	10:07:30
Kartlagt noe av informasjonen vi ønsker å hente ut og få vist	3:05:20
js/react + veiledning	3:05:59
Javascript tutorial by Derek Banas	1:09:29
Insert diverse data til customertabell	8:02:45
Innledning	10:00:31
Innføring i oracle application express. Testet mer på API	3:17:20
Innføring Apex, blitt kjent med dette	4:07:38
Implementasjon, database og struktur. Generell rapport	7:03:45
Generell rapportskriving	14:22:13
Forsøk på autentisering	8:01:43
Ferdigstilt update, refaktorisert noe kode til en "super update"	8:26:24
Ferdigstilt og testet update_device	2:07:40
Ferdigstilling av rapport	11:42:22
Ferdigstilling av prosjektplan	5:47:33
Fått update_device til å fungere	3:03:44
Diskusjon og utforming av grupperegler. Deligering av oppgaver for prosjektmal	5:30:00
Design, korrektur	5:34:09
Bestemme rammeverk, skaffe mere kunnskap av js	4:39:57
Avslutning, korrektur	10:06:56
Avklarte ting rundt struktur på applikasjon. Autentisering, oppdatering av data	3:09:48
APEX, hvordan legge inn json	7:21:26
APEX	12:29:03
Adde items til backlog, avklare uenigheter ifb. prosjektplan	2:59:16
3 step authentication i boks	7:47:10

Bjørn Ole Myrold	**377:10:00**
Veiledning	4:10:00
Utprøving av JS pakker	2:30:00
Usecase, Sekvensdiagram og rapportskriving	8:00:00
Usecase, raportskriving	3:00:00
Teknomøte med Evry	4:30:00
Statusmøte	1:00:00
Skriving av Prosjektplan	13:00:00
Siste småting prosjektplan	2:00:00
Rettlesning	2:00:00
Renskriving av møreref	1:00:00
Refresh_token og update_webrep	2:30:00
Refleksjon etter veil	1:00:00
React og basic layout	2:00:00
React	3:00:00
Rapportskriving, Sikkerhet, Usecase	4:30:00
Rapportskriving, refresh_token	11:30:00
Rapportskriving og lesing	10:00:00
Rapportskriving, hente data / oppdater database	4:00:00
Rapportskriving, database	10:00:00
Rapportskriving, calculate_risk	4:30:00
Rapportskriving	124:00:00
Raportformatertin, aka latex-magi	3:45:00
Prosjektrapport: Struktur og Innhold, samt APEX	4:00:00
Prosjektrapport: Struktur	3:30:00
Prosjektplan jobbing	1:00:00
Postmann, Aruba Central og smått web	4:30:00
Plan B	4:30:00
Møtevirksomheter, og kryptering	3:00:00
Møter	1:00:00
Møte med oppdragsgiver	1:00:00
Modelering	11:00:00
Litt API, mest React og basic layout	3:30:00
Kutte ned datamodel	8:00:00
Krypterings funksjonalitet	8:00:00
Kryptering, authcustomer	8:00:00
JavaScript / React	10:30:00
Introduksjon	3:30:00
Implementere datamodel	13:00:00

Helgajobbing	10:00:00
Gjennomgang av tidligere bachelor	3:00:00
Generell rapport	1:00:00
Gantt diagram	2:00:00
Fixeruppers prosjektplan	4:30:00
Fiksa på AUIR-trigger, refresh_token	5:30:00
Diverse møteaktiviteter	3:30:00
Div	1:00:00
Databaser	8:00:00
BM - Veiledning	0:30:00
BM - Oppstartsmøte Evry	2:00:00
BM - Oppstartsmøte	3:00:00
BM - Jobb	5:30:00
AuthCustomer funksjon ++	5:00:00
Aruba API	3:30:00
Apex	7:30:00
Angual JS	1:45:00

Created with toggl.com

H Møtereferat

H.1 Daily scrum/Dagslogg

H.1.1 Dagslogg 09.01.18

Forberedelser inn mot møte med oppdragsgiver
Møte med oppdragsgiver H.2.1

H.1.2 Dagslogg 11.01.18

Veiledningsmøte: 09:00, Møtereferat H.3.1 Har diskutert og underskrevet grupperegler. Fordelt arbeidsoppgaver i prosjektplanen. Levert kontrakt i BlackBoard.

H.1.3 Dagslogg 15.01.18

Forberedelser inn mot møte med oppdragsgiver og prosjektplan.

H.1.4 Dagslogg 16.01.18

Forberedelser inn mot møte med oppdragsgiver
Møte med oppdragsgiver H.2.2 Oppsett av Ruter hjemme, testing og bli kjent med Aruba

H.1.5 Dagslogg 24.01.18

Forbedringspotensiale på prosjektplan H.3.3

H.1.6 Dagslogg 29.01.18

Diverse valg av rammeverk: Node.js - for å bygge nettverksapplikasjon i JavaScript React - lage grensesnitt

Vurderte design til oversiktsbildet La til user stories med tilhørende tasks til backlogen

H.1.7 Daily scrum 05.02.18

Deltakere: Sondre, Guro, Bjørn Ole, Snorre
Sondre har brukt helga på å gå i gjennom Postman-verktøyet, og kjørte en kort gjennomgang for oss andre i teamet.
Videre ble vi enige om å bruke dagen på å gjøre oss godt kjent med dette verktøyet, og se om vi klarer å få kontakt med API-et ved ulike request-kall.

H.1.8 Daily Scrum 06.02.18

Deltakere: Sondre, Guro, Bjørn Ole, Snorre
Dagen før gjorde vi oss godt kjent med Postman, og vi klarte å opprette kontakt med API. Vi har per nå klart å hente ulik informasjon fra Aruba Central via både Postman og Aruba API-gateway(swagger).
Dagen i dag vil gå med på å fikse innlogging med både brukernavn og passord, samt teste ulike get-requests.

H.1.9 Daily Scrum 07.02.18

Deltakere: Sondre, Snorre
Dagen før klarte vi å liste alle devices(akksesspunkt og switcher), også per kunde. Vi mangler fortsatt å få determinert statusen på disse.
Bjørn Ole har fått laget et førsteutkast til login med tilhørende post-metode, dette skal vi blant annet bruke dagen til å se over og skjønne. Videre må vi avvente svar i fra Aruba support på en del spørsmål vi har om API-gatewayen.

H.1.10 Daily Scrum 08.02.18

Deltakere: Sondre, Bjørn Ole, Snorre
Dagen før gikk med på å lese seg opp på Bjørn Ole sitt arbeid, det viste seg forøvrig at dette ikke var fullstendig oppdatert, så det blir ny gjennomgang i dag. I tillegg leste vi oss opp på dokumentasjon om Aruba MSP og GET og POST.
I dag skal vi ha en gjennomgang av Oracle server med Nawar hos EVRY Norge AS, i tillegg tenker vi å definere noen spørringer og forklare url-oppsettet i rapporten.

Denne dagen ble vi stilt ovenfor et dilemma om vi skal bruke APEX eller ikke (APEX er fantastisk, men kan føre til at prosjektet blir for lett")

H.1.11 Daily Scrum 09.02.18

Deltakere: Bjørn Ole, Sondre, Snorre
Dagen før var vi hos Evry på Brummundal. Der fikk vi tilgang til og en kjapp innføring i et verktøy som heter Oracle Application Express(APEX), men vi har enda ikke fått mulighet til å benytte oss av Arubas Central REST-API, pga vanskeligheter med tilgang. Vi begynte også så smått med å strukturere rapporten vår, samt definere noen av spørringene vi anser som viktige.
Dagen i dag kommer til å gå med på å starte med litt rapportskriving, i tillegg tenker vi å legge til noen nye user-stories til backloggen vår.

H.1.12 Daily Scrum 12.02.18

Deltakere: Guro, Sondre, Snorre
Fredag forrige uke startet vi på innledningen på rapporten vår, i tillegg hadde vi en omfattende diskusjon rundt APEX-verktøyet og om dette kan være noe for oss.
Dagens agenda kommer til å omhandle å bringe Guro «up to speed» i prosessen så langt, samt lese oss mer opp på Oracle APEX så vi kan diskutere veivalget med veileder. Senere i dag skal vi ha retrospective-møtet vi skulle hatt på fredag i Guro sitt fravær. Under dette møtet skal vi definere sprint 2.

H.1.13 Daily Scrum 13.02.18

Deltakere: Sondre, Bjørn Ole, Guro, Snorre
I går brukte vi begynnelsen av dagen til å lese oss opp på og utforske Oracle APEX. Siste halvdel av dagen gikk med på møter, sprint retrospective, veiledning, og sprint planning.
I dag håper vi å endelig kunne bestemme oss for en utviklingsstack, denne avgjørelsen skal finne sted etter et møte med spesialister på APEX hos EVRY. Videre skal vi teste diverse funksjonalliter i APEX og bli bedre kjent med dette verktøyet.

H.1.14 Daily Scrum 14.02.18

Deltakere: Sondre, Snorre
Førdagen gikk med på å sette seg inn i Oracle APEX ved å aktivt teste diverse funksjonalitet, i tillegg fikk vi tilgang på et nytt workspace i versjon 5.1. Vi avventer fortsatt svar angående autentisering til API-et via riktig sertifikat.
Dagen i dag vil gå med på videre teste funksjonalitet rundt APEX.

H.1.15 Daily Scrum 15.02.18

Deltakere: Bjørn Ole, Snorre
Gårsdagen ble brukt til å sette seg inn i APEX sin måte å strukturere et view på. Vi hadde også et veiledningsmøte hvor vi fikk litt nøyere innføring i sertifikat og forslag til ting vi burde prøve å utforske mer på rundt Arubas API.
I dag ønsker vi å finne den konkrete forskjellen på MSP/Monitoring view i API-gatewayen, og hvordan få «tilgang» på Monitoring view. Vi ønsker også å se mer på funksjonaliteten rundt APEX, og kanskje se om vi klarer å få til sertifikatproblemet på egen hånd om vi får tid. Senere i ettermiddag skal vi ha et telefonmøte med Aruba support i California for å se om de kan hjelpe oss med våre gateway-problemer.

H.1.16 Daily Scrum 19.02.18

Deltakere: Guro, Sondre, Snorre
Forrige arbeidsdag, fredag 16.02 falt bort da ingen av oss var tilgjengelig. Vi skal derfor se på hva som ble gjort på torsdag 15.02.18.
Vi gjorde oss opp en forståelse om hva forskjellen mellom monitoring- og MSP view. I tillegg utforsket vi mer rundt funksjonaliteten i APEX og startet på litt placeholder design. Vi fikk dessverre ikke kontakt med vår kontaktperson hos Aruba.
I dag planlegger vi å fortsette videre på vårt placeholder design, og lage en mockup database som vi kan hente ut noen enkle men liknenede data som de vi typisk kan få fra Aruba. Videre er det relevant for de gruppemedlemene som trenger det å oppdatere sine SQL-kunnskaper, og sette seg inn i Oracle sin egen PL/SQL.

H.1.17 Daily Scrum 20.02.18

Deltakere: Sondre, Guro, Snorre
I går brukte vi hele dagen på å forstå informasjonsflyten fra server til database i APEX, og vi fikk til å legge inn noe mockup data i en tabell.
I dag kommer vi til å fortsette i samme leiet, med å forsøke å hente inn JSON til GUI ved å først legge det til en «collection» i APEX. Hvis det blir tid til overs, skal vi oppdatere rapporten med diverse problematikk vi har hatt.

H.1.18 Daily Scrum 21.02.18

Deltakere: Guro, Sondre, Snorre
I går fikk vi mange forskjellige gjennombrudd. Vi hadde en telefonsamtale med tech-support hos Aruba i California, og fikk dermed oppklart hvorfor vi ikke hadde tilgang på informasjon relevant for oss. Dette skulle nå rettes opp i ved å gi alle customers tilgang på API-gateway. Vi fikk også bekreftet at det vil bli satt opp en linux-server konfigurert for APEX, slik at vi får hentet ut Aruba info til APEX. På slutten av dagen fikk vi en innføring av en profesjonell ansatt hos EVRY rundt Oracle-databaser, gode praksiser og strukturen rundt systemet.

Dagen i dag vil gå med på et retrospective-møte og et sprint planning-møte. Vi skal også skrive en statusrapport.

H.1.19 Daily Scrum 26.02.18

Deltakere: Sondre, Guro, Snorre
Forrige gang vi var i aksjon, oppsummerte vi sprint nummer 2 og la planen videre for neste sprint. Vi skrev også en statusrapport som skal presenteres for veileder.
Denne dagen kommer til å gå med på å danne en struktur og oversikt over informasjon vi får hentet fra Aruba Central. Når dette er gjort kan vi begynne å modulere et førsteutkast for hvordan databasen vår skal se ut. Et siste punkt på agendaen blir å teste om API-kall fra en package i APEX går så plettfritt som vi håper.

H.1.20 Daily Scrum 27.02.18

Deltakere: Sondre, Guro, Bjørn Ole
I går ble det startet med moddelering av databasen. Vi prøvde å utføre API-kall fra PL/SQL og satt oss generelt mer inn i Oracle Databases og da også PL/SQL. Statusrapporten ble ferdigstilt.
Idag vil vi prøve å utvikle litt mer i PL/SQL og moddelere større deler av databasen. Dette vil baseres på kravspek-møte med Evry idag H.2.5.

Sondre, Bjørn Ole og Guro gjorde en avklaring rundt arbeidstimer og oppmøte. Tirsdag og Torsdag skal alle møte på Evry - her er sykmelding ok. Mandag og onsdag skal alle tilstrebe å jobbe på skolen i kjernetid 8-16, dersom dette ikke gjennomføres skal de andre i teamet få melding om dette så fort det lar seg gjøre. Her er det fast at Bjørn Ole har matte frem til kl 12. Ellers fortsetter vi å basere oppfølging av antall timer og arbeidsoppgaver på den enkelte.

H.1.21 Daily Scrum 28.02.18

Deltakere: Sondre, Guro
I går laget vi store deler av datamoduleringen. Vi fikk bekreftet at Enable API Gatewayer gjort av Aruba TAC og vi hadde Møte med Evry vedrørende krav og design av løsningen. Vi jobbet også noe med å overføre fra data. mod. til SQL og inn i APEX. I dag skal vi fortsette med databasemoddeleringa, lage PL/SQL ut av den og starte på å lage selve databasen med tanke på tables, sequences og triggere.

H.1.22 Daily Scrum 01.03.18

Deltakere: Sondre, Guro, Bjørn Ole
I går ble det jobbet en del med PL/SQL der man har startet med implementering av datamodellen for databasen. Det har også blitt sett på triggers og sequences i denne sammenheng.
I dag vil vi implementere databasemodellen i større grad, med de triggers og sequences som hører til. Altså vil dagen brukes til PL/SQL.

H.1.23 Daily Scrum 05.03.18

Deltakere: Guro, Sondre, Snorre
På slutten av forrige uke gjorde vi et forsøk på å autentisere fra package uten umiddelbart hell. Vi begynte også å bygge opp databasen og lagret unna noen data fra API-kall.
I dag skal vi fortsette fra der vi slapp. Andre arbeidsoppgaver blir bestemt og delegert ettersom de dukker opp.

H.1.24 Daily Scrum 06.03.18

Deltakere: Bjørn Ole, Sondre, Guro, Snorre
I går brukte vi dagen på å få til autentisering fra package gjennom PL/SQL. Vi fikk også nesten ferdigstilt databasen, og hentet noe data fra API som vi fikk parset slik at det i teorien snart skal kunne legges rett inn i databasen.
I dag skal vi fortsette med autentiseringen, ferdigstille databasen (her skal vi

først oppskalere for så å strippe av det som vi ikke trenger), og få lagt inn data fra API-et til denne.

H.1.25 Daily Scrum 07.03.18

Deltakere: Guro, Sondre, Snorre
I går fikk vi endelig autentiseringen til å fungere slik vi ønsker (med hardkodede verdier). Vi reffaktoriserte også databasen en smule.
I dag skal vi forsøke å legge anskaffet JSON fra API-et inn i databasetabellene våre, samt optimalisere autentiseringen.

H.1.26 Daily Scrum 08.03.18

Deltakere: Sondre, Bjørn Ole, Guro, Snorre
I går forsøkte vi oss fram med SQL insert for å få lagt til data i tabellene våre, uten videre hell. Vi hadde også en omfattende diskusjon rundt hele applikasjonen, og om hva slags handlinger og kall som skal skje når.
I dag skal vi følge opp med SQL insert, modelringsreffaktoriseringen. I tillegg skal vi begynne å eksperimentere med sikkerhet, hvordan kryptere autentiseringsinformasjonen vår.

H.1.27 Daily Scrum 09.03.18

Deltakere: Bjørn Ole, Guro, Sondre, Snorre
I går fikk vi ferdigstilt databasen vår. Vi utførte også en del generell databasekoding rundt det å legge inn data og oppdatere den. I tillegg hadde vi et meget produktivt samkjøringsmøte for å sørge for at alle har fiksert seg inn på samme mål.
Dagen i dag vil gå med på å oppsummere sprint 3, samt videre planlegging av sprint 4 og hvor vi ønsker å være før påskeferien.

H.1.28 Daily Scrum 13.03.18

Deltakere: Sondre, Bjørn Ole, Guro, Snorre
I går startet vi på en update-funksjon som skal oppdatere informasjonen i tabellene våre. Vi begynte også med å lage GUI for kundedetaljer og interaktiv håndtering av denne. Vi ble ferdig med autentisering av customer.
I dag skal vi fortsette med og forhåpentligvis bli ferdig med update-funksjonen og visning av kundedetaljer. Vi skal også lage en «add customer» funksjon, og avklare spørsmål rundt kryptering av databasen.

H.1.29 Daily Scrum 14.03.18

Deltakere: Guro, Sondre, Snorre
I går lagde vi ferdig SQL queries for visning av customer-data, for å kunne vise og editere innhold i tabellen. Vi fikk også koblet dette opp mot GUI. Vi startet også med å lage oppdateringsfunksjoner for noen tabeller, og triggere for tabellen customer-details. Tiden gikk også med på å implementere kryptering/dekrypteringsnøkler.
I dag kommer vi til å fortsette med å ferdigstille de update-funksjoner og triggere vi har startet på.

H.1.30 Daily Scrum 15.03.18

Deltakere: Bjørn Ole, Guro, Snorre
I går gikk tida vi hadde med på å videreutvikle de update-funksjonene og triggerene vi allerede har startet på.
Dagen i dag vil gå med på å ferdigstille disse, prøve å finne licensing, jobbe videre med krypto, samt lage en funksjon for å refreshe token.

H.1.31 Daily Scrum 19.03.18

Deltakere: Sondre, Guro
Sist gang ble «TRG_AWS_CUSTOMER_DETAIL_AIUR» nesten ferdig. Startet

på updateDevice-funksjon. Authcustomer funksjoner ferdigstillt. Resfreshtoken ble nesten ferdig. Kryptering fungerende, og tabeller for å holde krypteringsnøkler er på plass.
I dag skal vi fikse AIUR-trigger. Oppdatere hele hierarkiet mellom customer detail, customer, customer key ved hjelp av AIUR-trigger.

H.1.32 Daily Scrum 20.03.18

Deltakere: Sondre, Guro
Sist gang ble «TRG_AWS_CUSTOMER_DETAIL_AIUR» ferdig. Fortsatte på updateDevice og updateNetwork-funksjon. Resfreshtoken ble ferdig. Kryptering fungerende, og tabeller for å holde krypteringsnøkler ble testet og satt inn der token har vært hardkodet.
I dag skal vi legge inn flere update funksjoner i AIUR-trigger. Oppdatere hele hierarkivet mellom customer detail, customer, customer key ved hjelp av AIUR-trigger og teste dette.

H.1.33 Daily Scrum 22.03.18

Deltakere: Sondre, Guro, Bjørn Ole, Snorre
I går ble arbeidet noe amputert pga fravær.
Sist gang ble hele hierarkiet mellom customer detail, customer, customer key ved hjelp av AIUR-trigger og ut i GUI fungerende og er under testing.
Startet på updateDevice-funksjon. Authcustomer funksjoner ferdigstillt. Resfreshtoken ble nesten ferdig. Kryptering fungerende, og tabeller for å holde krypteringsnøkler.
I dag skal vi fikse AIUR-trigger. Oppdatere hele hierarkiet mellom customer detail, customer, customer key ved hjelp av AIUR-trigger.

H.1.34 Daily Scrum 03.04.18

Deltakere: Guro, Sondre, Snorre
Før påske fikk vi sørget for at et par av update-funksjonene våre fungerer som de skal, og at uthentete krypterte tokens blir dekryptert på riktig måte. Vi fikk også ferdigstilt AIUR-triggeren vår.

I dag vil gå med på å ferdigstille alle update-funksjoner så langt det rekker. Vi har i tillegg planer om å teste ut scheduled jobs for å få startet på fasen hvor vi syr sammen delene i programmet vårt.

H.1.35 Daily Scrum 04.04.18

Deltakere: Guro, Snorre
Gårsdagen ble brukt til å ferdigstille alle update-funksjoner. Vi fant også ut at vi må lage en «super» update som kjører alle andre updates for å koble denne mot en scheduled job. Vi har også dessverre oppdaget at det er blitt foretatt noen endringer for autentisering mot API-et vi bruker, noe som fører til at vår nåværende løsning ikke fungerer lenger.
Vi kommer til å bruke timene vi har i dag på å få «super-prosedyren» til å fungere, samt feilsøke vårt nyoppståtte problem.

H.1.36 Daily Scrum 05.04.18

Deltakere: Bjørn Ole, Sondre, Guro, Snorre
I går brukte vi tid på å feilsøke problemet med autentisering, men kom ingen vei med dette foreløpig. Vi har også fått lagt inn en scheduled job på refreshing av MSP-token, i tillegg jobbet vi videre med update funksjonen som etterhvert skal kobles mot en scheduled job.
I dag skal vi feilsøke autentiseringsproblemet videre og jobbe med scheduled job mot update.

H.1.37 Daily Scrum 09.04.18

Deltakere: Sondre, Snorre
I løpet av helgen fikk vi fikset opp i autentiseringsproblemet vårt, noe som betyr at vi er tilbake på rett spor. Nå begynner det å nærme seg en fungerende minimumsløsning, men noen småting mangler.
I dag skal vi prøve å fikse refresh-tokens prosedyren vår, slik at dette forhåpentligvis snart skal gå automatisk. I tillegg er vi i tvil om device-kallene henter ut korrekt antall, derfor skal denne også reviewes på nytt.

Rapport:
Forrige gang: Snart ferdig med førsteutkast til innledning, snart ferdig med use-case.
I dag: Gjøre ferdig innledning og use case.

H.1.38 Daily Scrum 10.04.18

Deltakere: Bjørn Ole, Sondre, Snorre
I går brukte vi dagen på å få refresh tokens til å fungere, samt foreta SQL-queries mot GUI for å vise det vi ønsker.
I dag kommer vi til å foulproofe refresh tokens, reviewe update funksjoner og sette opp schedulere.

Rapport:
Forrige gang: Førsteutkast og til innledning og use-case i kravspec er klar for dom av veileder.
I dag: Se over og sende utkast til veileder. Fortsette på kravspec og generelt rapport.

H.1.39 Daily Scrum 11.04.18

Deltakere: Sondre, Snorre
I går fullførte vi de oppgavene som vi hadde satt oss.
I dag skal vi kun stikkteste schedulerene våre.

Rapport:
I går gikk alle gruppemedlemer gjennom det som er blitt skrevet så langt. Vi fikk også lagt flere relevante figurer til dokumentet.
I dag skal vi starte på «metode»-kapitelet, samt skrive mer om spesifikasjoner.

H.1.40 Daily Scrum 12.04.18

Deltakere: Guro, Bjørn Ole, Sondre, Snorre
I går fikset vi scheduled job på refresh token, samt jobbet vi videre med SQL-queries mot GUI.

I dag skal vi fortsette med sistnevnte, i tillegg skal vi jobbe videre med web rep.

Rapport:
I går skrev vi videre på spesifikasjonskapittelet, vi begynte også på metode om valg av utviklingsstack.
I dag skal vi fortsette med å utbedre use case modellen vår, lage et sekvensdiagram og en innledning til spesifikasjonskapittelet.

H.1.41 Daily Scrum 16.04.18

Deltakere: Sondre, Snorre
Forrige gang ble det jobbet med web-reputation og SQL mot GUI.
I dag skal vi fortsette med begge overnevnte, da vi ikke klarte å ferdigstille noen av dem. I tillegg skal vi gå tilbake på network-tabellen for å se om vi klarer å få lagt til riktige verdier i bandwidth-kolonna.

Rapport:
Forrige gang rettet vi litt opp i use-casene våre etter gode input fra Guro som er studentassistent i emnet systemutvikling. Vi fikk også begynt på metodekapitelet og har fått skrevet litt om de valg og fremgangsmåte vi har hatt.
I dag kommer vi til å skrive mer om metodikk, samt reviewe spesifikasjonkapittelet for vurdering.

H.1.42 Daily Scrum 17.04.18

Deltakere: Bjørn Ole, Guro, Sondre, Snorre
I går ble bandwidth-problematikken løst, og vi får nå hentet inn riktig verdi og konvertert denne til epoch-format. Vi jobbet også videre med web-rep, men denne sliter vi fortsatt med. Videre jobbet vi med GUI, men dette er heller ikke helt ferdig.
I dag vil vi fortsette med web-rep og GUI.

Rapport:
I går gjorde vi ferdig et førsteutkast for kapittel 2, spesifikasjoner. Vi jobbet også med kapittel 3, og har kommet et lite stykke på vei med dette. Midt på dagen hadde vi et planleggingsmøte angående hva som skal være med i

rapporten.
I dag skal vi skrive videre på kapittel 3, og muligens begynne på en brukermanual.

H.1.43 Daily Scrum 18.04.18

Deltakere: Guro, Snorre, Sondre
I går ble bandwidth ferdigstillt(viser MegaBit). Refreshing av nettsiden hvert 5 min ferdig implementert. Guro jobbet med log til fil, men vi tenker å logge til en tabell i stedet.
I dag skal vi se mer på hvordan vi enkelt kan internasjonalisere overvåkingssystemet, i tillegg til logging.

Rapport:
I går skrev vi videre på implementasjon, og fikk ordnet noen nye definisjoner på ting som vi så trengte det. Vi gjorde også litt om på introduksjonskapittelet for å bedre strukturen og gi leseren bedre forståelse for hva som blir omtalt senere.
I dag skal vi skrive ferdig «implementasjon iht krav», og skrive litt mer rundt API-et. Vi skal også ha et veiledningsmøte angående rapporten så langt.

H.1.44 Daily Scrum 19.04.18

Deltakere: Bjørn Ole, Guro, Snorre
I går fant vi en grei måte å internasjonalisere på, ved å bruke eksisterende funksjonalitet i APEX kan vi lage oversetning for all tekst, og så trykke på en knapp for å få det oversatt.
I dag skal vi kun fokusere på rapport, og utbedre noen av punktene som ble omtalt på veiledningsmøte.

Rapport:
I går skrev vi en brukermanual for overvåkingssystemet og en mer fyldig beskrivelse av implementasjon.
I dag skal vi skrive mer rundt hvordan oppdatering av databasen foregår, i tillegg til hvordan API-et brukes og hvordan man utfører kall mot dette.

H.1.45 Daily Scrum 23.04.18

Deltakere: Sondre, Snorre
I dag skal vi prøve å få fullført logging av databasen.

Rapport:
Forrige gang ble det skrevet om hvordan databasen oppdateres, kall mot API-et, og generelt om databasen.
I dag skal vi i bunn og grunn fortsette der vi slapp, og forklare nærmere hva de ulike tabellene i databasen inneholder, deres funksjonalitet og relasjoner.

H.1.46 Daily Scrum 24.04.18

Deltakere: Guro, Sondre, Snorre
I går fikk vi ferdigstilt loggingen.
I dag skal vi teste den nye kontoen vi har opprettet for systemet. Om det blir tid skal vi også review web-reputation.

Rapport:
Gårsdagen gikk med på mye rettskriving og omformulering. Implementasjonskapittelet er snart ferdigstilt, og det ble så vidt påbegynt arbeidsprosess.
I dag skal vi fortsette på kapittelet om arbeidsprosess og skrive om noen av de utfordringer og realiseringer vi har hatt undervegs, samt renskrive en del møtereferater.

H.1.47 Daily Scrum 25.04.18

Deltakere: Snorre
I går fikk vi testet den nye kundekontoen, og den fungerte med den hensikten som var planlagt. Vi gikk også gjennom web-reputation en siste gang, og har endelig konkludert med at denne ikke gjøres noe mer med da vi ikke får lagt inn den informasjonen som er ønskelig.

Rapport:
I går ble møtereferatene finskrevet slik at de gir et bedre overblikk over sammenheng.
I dag skal det skrives om arbeidsprosessen.

H.1.48 Daily Scrum 26.04.18

Deltakere: Bjørn Ole, Sondre, Guro, Snorre
I går ble ordnet opp i en job som hadde forårsaket en «deadlock».
I dag skal vi sjekke at kundene i visningen sorteres på riktig vis, finne relasjoner mellom brukere og database for å sikre at det kun er Aruba-brukeren som har tilgang til krypteringsnøklene. Videre skal vi gå igjennom koden og kommentere mer utfyllende og forklarende.

Rapport:
I går startet vi å skrive om arbeidsprosessen vår undervegs og de problemer vi har støtet på, og erfaringer vi har gjort oss.
I dag skal fortsette der vi slapp, og rette litt i teksten etter innspill fra gårsdagens veiledning. Vi skal også rette på use case og sekvensdiagram, samt lage noen arkitekturmodeller dersom det blir tid.

H.2 Møtereferat: Oppdragsgiver

H.2.1 Kontraktsmøte med EVRY

Dato: 09.01.18
Agenda: Underskrive kontrakt, innblikk i EVRY Norge AS som bedrift, hvordan løsningen fungerer i dag.
Sted: EVRY Norge AS, Brumunddal
Varighet: 90 minutter

Ved møtestart fikk gruppen et grundig innsyn i EVRY Norge AS som bedrift, hva slags tjenester de tilbyr sine kunder, antall ansatte og organisasjonskart.

Videre fikk gruppen en innføring i hvordan løsningen fungerer i dag, og hvorfor oppdragsgiver ikke er fornøyd med dagens funksjonalitet. Gruppen fikk en overordnet kravspesifikasjon om behov og hva den nye løsningen burde inneholde, blant annet en ny kundeoversikt som ikke skulle ha behov for alt for mye navigering, og et bedre varslingssystem, slik at responstid mot kunde går raskere

Til slutt ble det diskutert, og skrevet under på taushetsplikt, eiendomsrett,

og generelle forholdsregler.

H.2.2 Teknisk oppstartsmøte med EVRY

Dato: 16.01.18
Agenda: Innføring i NaaS som tjeneste, teknisk rundt Aruba Central og dokumentasjon, oppsett på egne pcer med Aruba-konto.
Sted: EVRY Norge AS, Brumunddal
Varighet: 120 minutter

Under dette møtet fikk gruppen en innføring i hva NaaS innebærer, og hvordan prosessen i å bistå kundene under denne tjenesten foregår. Det ble holdt en presentasjon om hvordan teknisk spesialist møter fysisk opp hos kunde, og installerer aksesspunkt og andre nettverksenheter etter lokasjon, bygningsmasse og romplassering.

Videre ble det gitt en innføring i Aruba Central sin dokumentasjon, og hvordan benytte seg av deres API-gateway. Gruppemedlemmene fikk også lage sine egne kontoer for å få tilgang til systemet. Med dette fikk gruppen dannet seg et oversiktsbilde om hva som skulle jobbes med videre. Det ble også formulert videre en kravspesifikasjon mer spesifikt rundt hva oppdragsgiver ønsket at selve løsningen skulle inneholde av informasjon i hovedvisningen.

Avslutningsvis ble det oppsummert med videre plan og bistand fra oppdragsgiver, hvor det blant annet ble satt opp et møte med en ansatt hos oppdragsgiver som har jobbet med løsningen før. Det ble også avtalt at gruppen skulle ha et rom tilgjengelig tirsdag og torsdag under hele utviklingsperioden.

H.2.3 Møte med EVRY-ansatt

Dato: 01.02.18
Agenda: Informasjon rundt hva som er blitt prøvd før, introduksjon av Postman.
Sted: EVRY Norge AS, Brumunddal
Varighet: 40 minutter

Under dette møtet fikk gruppen informasjon om hva som er blitt prøvd i forbindelse med det å få kontakt med API-et, og forslag til fremgangsmøte

for å få bukt med denne problematikken.

Gruppen ble også introdusert til et verktøyet postman. Dette kunne tas i bruk for å teste API-kall mot Aruba Central, hvor man enkelt kunne se innholdet i header, body, og respons for hvert kall. Det ble også utvekslet noen url-eksempler for autentisering mot API-et, samt noen GET-requests for å hente ut informasjon om blant annetaksesspunkt.

H.2.4 Møte 2 med EVRY-ansatt

Dato: 08.02.18
Agenda: akkreditiver for innlogging + innføring i Oracle APEX.
Sted: EVRY Norge AS, Brumunddal
Varighet: 20 minutter

Dette møtet gikk ut på å introdusere gruppen til Oracle APEX, og hvordan få tilgang til dette. Gruppen fikk mulighet til å utforske verktøyet, og vurdere om dette var noe de ønsket å ta i bruk.

H.2.5 Kravspec-møte med EVRY

Dato: 27.02.18
Agenda: Konkret kravspesifikasjon.
Sted: EVRY Norge AS, Brumunddal
Varighet: 60 minutter

Bjørn Kråkevik ledet møtet med spesifikke konkrete krav. Han tegnet opp hvordan de ville ha visningen med en tabell med kundene vertikalt og diverse info horisontalt. I disse kolonnene skulle det informasjon om hvor mange aksesspunker og switcher som er i og ute av drift til enhver tid. Hvor mye data/trafikk som går til en kunde og hva slags «Network Health» nettverkstrafikken hadde skulle også med i tabellen. Tom Christensen ville også at ROGUE AP(eksterne aksesspunkt som ikke tilhører nettverket) skulle komme i tabellen hvis det var noe vi fikk tid til.

Rekkefølgen på kundene skulle vises med de kundene med mest kristisk feil øverst. En enkel fargelegging med for eksempel rødt ble også ønsket for å markere unntak.

H.3 Møtereferat: Veiledning

H.3.1 Veiledningsmøte 1

Dato: 11.01.18
Agenda: Framtidige veiledningsmøter, forprosjekt, generelt rundt oppgaven.
Sted: NTNU Gjøvik
Varighet: 40 minutter

Dette møtet ble brukt til å planlegge når framtidige veiledningsmøter skulle finne sted, hvor det ble bestemt onsdager klokken 12:15, om ikke manglende behov fra gruppen, eller andre omstendigheter forhindret dette. Det ble også diskutert hva som var forventet av forprosjektet som skulle leveres innen 01.02.

Videre ble det pratet om selve oppgaven, og gruppen fikk noen gode innfallsvinkler på hvordan tilnærme seg oppgaven i oppstartsprosessen.

H.3.2 Veiledning 2

Dato: 17.01.18
Agenda: Utviklingsmiljø/plattform, tilbakemelding på prosjektplan
Sted: NTNU Gjøvik
Varighet: 30 minutter

Noe som var viktig for gruppen å få avklart relativt tidlig, var hva slags rammeverk, utviklingsmiljø og eventuelt plattform som skulle tas i bruk. Gruppen fikk diskutert dette med veileder, og fikk noen innspill på hva som burde vektlegges i valget.

Det ble også gitt tilbakemelding på strukturen av prosjektplanen, og ting som var viktig å fastslå tidlig for å ha et best mulig grunnlag for den tid hvor utviklingen skulle starte.

H.3.3 Veiledning 3

Dato: 24.01.18
Agenda: Forbedringspotensiale på prosjektplan
Sted: NTNU Gjøvik
Varighet: 30 minutter

Dette møtet gikk med på å diskutere oppsettet på prosjektplanen så langt. Gruppen fikk tilbakemelding på at prosjektmålene var litt tynne, men at struktur og referanser generelt så godt ut.

H.3.4 Veiledning 4

Dato: 31.01.18
Agenda: Finpuss på prosjektplan
Sted: NTNU Gjøvik
Varighet: 30 minutter

Gruppen hadde problem med å utforme en «tradisjonell» problemstilling, da dette prosjektet er noe annet enn hva meldemmene har dokumentert før. Det viste seg å ikke være så veldig vanskelig, og medlemmene fikk formulert en god problemstilling allerede under møtet.

Videre ble det rettet opp i noen flere småting rundt planen, samt gitt noen tips til elementer som var viktig å holde en knapp på allerede fra utviklingsstart, som for eksempel konsekvent fordeling av arbeidsoppgaver, testing og sikkerhet.

H.3.5 Veiledningsmøte 5

Dato: 15.02.18
Agenda: Avvik fra initiell prosjektplan, hjelp til sertifisering
Sted: NTNU Gjøvik
Varighet: 40 minutter

Utviklerne opplevde veldig tidlig i utviklingsprosessen å bli introdusert til et nytt rammeverk, med et helt annet programmeringsspråk til bruk en det som initielt var planlagt. Det var dermed et sterkt behov på innspill om hva

som burde velges, og hvordan formulere i rapporten en eventuelt såpass stor endring fra den opprinnelige planen.

Utviklerne kom til enighet med veileder om at det var best å velge et rammeverk med mange integrerte moduler for utvikling, da det er det som er mest vanlig å benytte i arbeidslivet. I tillegg var det oppfattet som smart å velge et rammeverk som oppdragsgiver allerede var kjent og vandt med, slik at det kunne bli enklere for dem å bistå utviklerne videre under prosjektet.

H.3.6 Veiledningsmøte 6

Dato: 28.02.18
Agenda: Problemer rundt Aruba Central REST-API
Sted: NTNU Gjøvik
Varighet: 30 minutter

Gjennom den første utviklingsmåneden møtte utviklerteamet på flere utfordringer rundt det å få kontakt med API-et, få lagt inn rett sertifikat for å opprette kommunikasjon mellom Oracle APEX og Aruba Central, og det å forstå de ulike URL-ene, og hva de returnerte i de ulike responsene. Dessverre var utviklerne helt avhengig av teknisk support fra Aruba, noe som betød mange mail-korrespondanser med mye venting.

Veileders oppfatning, var at utviklerne har vært flinke på å arbeide videre med andre ting, selv om prosjektet til tider har manglet progresjon på grunn av det overnevnte. Han ga videre innspill til hvordan dette burde formuleres i rapporten.

H.3.7 Veiledingsmøte 7

Dato: 04.04.18
Agenda: Struktur på prosjektrapport, utviklingsarbeidet så langt
Sted: NTNU Gjøvik
Varighet: 45 minutter

Dette var det første veiledningsmøte siden 28.02, og utviklerne hadde på dette tidspunktet fått ferdigstilt mye av løsningen. Fokuset ville nå begynne

å flyttes over på rapportskriving, og gruppen trengte noen innspill på hvordan strukturen kunne være.

Da dette var gjort, ble det snakket om utviklingsarbeidet så langt, og utviklerne fikk lagt fram at Aruba hadde endret på autentiseringsinformasjonen sin over påske, grunnet et sikkerhetshull utviklerne trolig selv hadde oppdaget. I tillegg var mange av JSON-verdiene også endret, slik at løsningen ikke lenger fikk oppdatert databasen som visningen henter informasjon fra. Veileder poengterte at dette var noe utviklerne måtte tilpasse seg, og passe på å dokumentere godt i tilfelle noe liknende skulle oppstå igjen.

H.3.8 Veiledingsmøte 8

Dato: 11.04.18
Agenda: Tilbakemelding på innledning av rapport
Sted: NTNU Gjøvik
Varighet: 30 minutter

Under dette møtet fikk gruppen tilbakemelding på innledningskapittelet i prosjektrapporten, og forslag til forbedringer. Strukturen begynte å se bra ut, men det var fortsatt noe forbedringspotensiale rundt hvordan ting hang sammen.

H.3.9 Veiledningsmøte 9

Dato: 19.04.18
Agenda: Tilbakemelding på spesifikasjon og implementasjon, kryptering (rapport)
Sted: NTNU Gjøvik
Varighet: 45 minutter

Her fikk gruppen tilbakemelding på den mer tekniske biten av rapporten, og hvordan best mulig få frem hva som er blitt gjort i forhold til utviklingsprosessen. Det ble i tillegg diskutert usikkerheten rundt krypteringsprotokollen mot Aruba, om det fortsatt ble kjørt på SSL. Dette måtte gruppen få avklart snarest.

H.3.10 Veiledningsmøte 10

Dato: 25.04.18
Agenda: Tilbakemelding på implementasjon, sikkerheten i produktet
Sted: NTNU Gjøvik
Varighet: 45 minutter

I dette møtet fikk gruppen videre veiledning på rapportens struktur og innhold, samt en oppklaring i forståelse av tilkoblingen til API-et og sikkerheten i databasen. Veileder påpekte enkelte potensielle hull i sikkerhetsimplementeringen til systemet, og oppfordret gruppen til å sikre at tilgang til kritisk informasjon kun kunne aksesseres av systemet selv.

H.3.11 Veiledningsmøte 11

Dato: 02.05.18
Agenda: Rapportstruktur
Sted: NTNU Gjøvik
Varighet: 30 minutter

Under møtet ble hovedsaklig kapittel om implementering snakket om. Herunder strukturelle og innholdsmessige kommentarer for eks på hente data, testing og kildekodelisting. Noe overordnet om helheten ble også diskutert for eks vedlegg og kildekode, samt "hvordan vi ligger ann".

H.3.12 Veiledningsmøte 12

Dato: 09.05.18
Agenda: Rapportstruktur
Sted: NTNU Gjøvik
Varighet: 50 minutter

I dette møtet så vi overordnet på strukturelle elementer. Det er ingen flere store endringer i innhold som skal gjøres. Det ble påpekt av veileder at det er viktig å lese over og gjøre eventuelle endringer med tanke på den røde tråden. Fint å printe ut rapporten mot slutten også for å sjekke figurer mm. Eller småpirk som ble tatt opp.

H.4 Sprint retrospective

H.4.1 Sprint 1: 12.02.18

Hva gikk bra

Det første vi er fornøyd med, er at vi klarte å få kontakt med API-et innen rimelig tid, og at vi klarte både å få autentisert oss og listet diverse interessant informasjon. Videre var vi veldig fornøyd med at vi har klart å utnytte de ressursene som er tilgjengelig for oss; dette innebærer personell hos oppdragsgiver som har bistått med innføring i programmer som har vært til nytte for oss, slik som Postman og Oracle APEX. I løpet av Sprint 1 har vi klart å utføre alle de arbeidsoppgavene som vi har delegert til denne sprinten, selv om vi innser at vi burde hatt flere backlog-items å ta av. Mot slutten av sprinten fikk vi også tid til å starte med struktur og skriving av rapporten, noe som kommer godt med til senere rapportskriving.

Hva gikk dårlig og hvordan forbedre

Vi innser at vi er alt for dårlig videreføring av viktig informasjon til hele gruppa. Vi skal klare å forbedre oss ved å ha økt fokus på dette ved å notere i fila «Kladd Utvikling», samt fysisk si i fra til alle i utviklerteamet.
Vi overestimerte arbeidsmengden, så vi kunne hatt mer å gjøre mot slutten av sprint 1. Det at vi var mye usikre på hva vi faktisk skulle gjøre medførte også til litt dårlig arbeidsflyt. Neste sprint skal vi lage flere og mer konkrete PBI, og legge større vekt på å forstå helheten.
Vi hadde noe dårlig ansvarsfordeling, dette skal vi løse ved å bruke "Assign to"i Taiga, da vil ansvaret fordeles i større grad til enkeltpersoner.

H.4.2 Sprint 2: 21.02.18

Hva gikk bra

Vi er veldig fornøyde med måten vi har gjennomført forbedringstiltakene fra forrige sprint på. Vi føler at kommunikasjonen gruppa i mellom er blitt

markant bedre. Videre er vi blitt mye flinkere til å formulere spørsmål rettet mot Aruba TAC, slik at responstiden er blitt redusert. Vi har dermed også fått bekreftet en del teorier vi hadde rundt API-gatewayen. Vår kompetanse med Oracle APEX er også merkbart utvidet takket være egen innsats og mye god hjelp og veiledning fra fagpersonell hos EVRY Norge AS.

Hva gikk dårlig og hvordan forbedre

Elefanten i rommet under sprint 2 har vært de olympiske vinterleker. Dette har tatt opp en grei porsjon av arbeidstiden. En annen ting som er nødvendig å sette fingeren på er at vi har vært alt for dårlig på å finne alternative ting å gjøre i tider vi har ventet på svar på Aruba TAC og dermed «stanga» mye. Dette hovedsaklig grunnet at vi har vært alt for dårlig på å følge vår egen plan i Taiga. Hovedpunktet for forbedring til neste sprint vil være å bli enda flinkere på å legge inn enda mer konkrete kort i Taiga, og følge disse

H.4.3 Sprint 3: 09.03.18

Hva gikk bra

Vi føler selv at vi har blitt meget bedre på å sørge for felles forståelse innad i gruppa, både når det kommer til delmål og hva alle forsøker å utføre til enhver tid. Vi har også hatt god fremgang, og det er en del ting som har begynt å løsne for oss, slik at vi nå har en god oversikt på hvor langt vi klarer, og ønsker å komme. Vi har også hatt en mye bedre arbeidsflyt grunnet tiltakene vi igangsatte etter sprint 2, med å være enda mer konkrete på arbeidsoppgaver og hvordan vi delegerte disse. Vi kan med trygghet si at dette har vært vår beste og mest effektive sprint så langt.

Hva gikk dårlig og hvordan forbedre

Vi klarte ikke å fullføre alle oppgavene som var lagt opp i taiga-tavla vår, så vi feilestimerte litt på arbeidsmengden. Dette er det bare å fortsette å forsøke å bli bedre på.
Denne sprintperioden var også noget sykdomsutsatt, hvor Snorre var borte

i fire dager i strekk grunnet influensa. Dette taklet gruppa greit og arbeidsprosessen ble ikke hemmet i noen større grad. En siste ting vi har innsett er at vi må øve mer på å lese dokumentasjon for å finne de helt riktige og konkrete tingene vi er ute etter.
Et viktig tiltak for neste sprint blir å konkretisere og avgrense oppgavene videre enda mer, slik at vi er sikker på å ha på plass det vi ønsker før vi tar påskeferie.

H.4.4 Sprint 4: 22.03.18

Hva gikk bra

Vi følte vi hadde god progresjon til tross for noe uventet fravær, og vi har blitt mye bedre til å dokumentere utført arbeid, samt videreformidling av informasjon til gruppemedlemmer som har vært fraværende. En annen ting som vi har hatt litt dårlig utførelse på tidligere, er det å lete fram korrekt dokumentasjon for gitte problemstillinger. Dette er nå blitt et mye mer begrenset problem, og vi trenger ikke å konsultere oss med fagpersonell i like stor grad som før.

Hva gikk dårlig og hvordan forbedre

Vi greide dessverre ikke å nå den målsettingen vi hadde satt oss i begynnelsen av denne sprinten, og vi ble hengende etter på taiga-tavla vår. Dette antar vi er mye grunnet det uventede fraværet fra flere av gruppemedlemmene.
Vi er blitt oppmerksom på at uventet fravær har vært årsak til manglende gjennomførelse av arbeidsoppgaver i to sprinter på rad nå, så vi ser en sammenheng i dette og kan dra den antagelsen at våre estimeringer ikke er så feil som planleggingsprosessen kanskje viser.

H.4.5 Sprint 5: 16.04.18

Hva gikk bra

I denne sprinten var det minimalt med fravær fra samtlige gruppemedlemmer, og ingenting av nevnt fravær har vært sykdom. Dette resulterte i at vi denne gangen traff godt på estimering av arbeidsmengde, noe vi er veldig fornøyd med, og det bekrefter i stor grad antakelsen vi noterte oss i forrige retrospective-møte. Videre er vi veldig godt fornøyd med at vi nå har en fungerende minimumsløsning, slik at siste sprint kan gå med på å optimalisere denne. Vi har i tillegg også hatt meget godt samarbeid gjennom denne sprinten, kommunisert godt og delegert arbeidsoppgaver på en fornuftig og hensiktsmessig måte.

Hva gikk dårlig og hvordan forbedre

Internt i gruppen er det ingenting som har gått så dårlig under denne sprinten at det er verdt å nevne, utenom helt vanlig småuenigheter angående gjennomførelse. Hovedproblemet i denne sprinten oppstod helt i starten hvor vi kom tilbake i arbeid etter påske, for å finne ut at aruba hadde endret sine akkreditiv for å kunne autentisere for tilgang til API-et, samt endret på noen av navnene til JSON-verdiene. Problemet ble raskt ordnet opp i så snart vi fant ut hva som hadde skjedd, men vi må også ta til ettertanke at noe sånt kan skje igjen, slik at vi kan gjøre brukerne av systemet oppmerksom på hva som må rettes opp i et slikt tilfelle. Som forbedring skal vi fokusere enda mer på å bli flinkere til å bruke hverandre som hjelpeverktøy.

H.4.6 Sprint 6: 27.04.18

Hva gikk bra

Har klart å lage et relativt ferdig produkt og gjorde det til den interne fristen som var satt til ut denne sprinten(27.04.18). Ressursfordelingen ved at noen jobbet på det siste av utvikling og resten på rapportskriving har vært bra, ble ferdig produkt og samtidig gjort mye på rapport.

Hva gikk dårlig

Enkelte av funksjonaliteten ble ikke implementert(web reputation på et nettverk). Testing av systemet ble ikke gjort. Forbedringen her er å være flinkere til å planlegge og legge inn i backlog.

I Statusrapporter

I.1 Statusrapport 21.02.18

I.1.1 Fremdrift

Prosjektets tilstand er bra og fremdriften har vært noe bak forventet, men er på tur til å ta seg opp. Vår største risiko var å ikke få kontakt og korrekt informasjon fra Aruba tidlig i prosessen. Vi har nå fått kontakt med APIet hos Aruba Central via APEX og får hentet ut det meste av informasjon. Vi har antagelig funnet ut hvorfor vi ikke får den informasjonen vi ønsker, men dette skal snart være i orden.

Vi hadde forventet å komme noe lengre i utviklingen per dags dato, men vi ser i etterkant at arbeidsoppgavene vi har gjennomført har vært naturlige prosesser for å lage en så god løsning som mulig.

I.1.2 Avvik

Vi har gjort en endring i valg av standard verktøy. Vi hadde i utgangspunktet planlagt å utvikle løsningen i JavaScript, NodeJS og React, men etter at vi fikk tilgang til Oracle APEX av EVRY, valgte vi å endre utviklingsplattform til den platformen oppdragsgiver tilbyr og er kjent med. Vi gjorde en vurdering på om dette valget ville medføre en sterk nedgang i egenprodusert kode, og om dette ville ha en negativ effekt på bacheloroppgaven.

Vi ser at APEX vil tilføre prosjektet stor verdi ved at det gjør Front-End arbeidet mye enklere. Teamet valgte derfor å endre plattform, slik at vi heller kunne bruke tid innspart til å utvikle løsningen videre. Avgjørelsen så vi også som riktig, da APEX er et produkt av nyere teknologi og legger tilrette for mindre «slavearbeid» i utviklingen.

I.1.3 Konklusjon

Arbeidet er veldig lærerikt og teamet er meget motivert. Vi har møtt noen utfordringer undervegs, men har lært av de og kommet videre fra disse. Teamet jobber offensivt og har stor tro på både utviklingen og oppgaven.

I.2 Statusrapport 23.03.18

I.2.1 Fremdrift

Prosjektets tilstand er bra og fremdriften har tatt seg godt opp siden siste statusrapport. Vi har nå fått kontakt med API-et hos Aruba Central via APEX og får hentet ut det informasjon både som MSP-bruker (Master) og Customer-bruker (Detaljert).

Vi hadde forventet å komme noe lengre i utviklingen per dags dato, men vi ser i etterkant at arbeidsoppgavene vi har gjennomført har vært naturlige prosesser for å lage en så god løsning som mulig.

Vi hadde en teori ved forrige statusrapport om at problemet vårt var at en tilgang ikke var aktivert. Teorien vår var korrekt og problemet ble borte når dette ble aktivert.

I.2.2 Avvik

Vi har gjort en endring i valg av standardverktøy. Vi har blant annet satt opp databasen via SQL Developer noe som var et ukjent program for oss fram til det ble introdusert i starten av sprint 3. SQL Developer har en dårlig versjonskontroll og vi valgte derfor å fortsatt ha versjonskontroll hos Bitbucket og bruker Atom for å utføre dette. Vi har også hatt noe fravær denne perioden pga sykdom og andre uforutsette hindringer. Vi har hatt gode rutiner slik at dette ikke har medført store problemer, men det ble noe tapt utviklingstid.

I.2.3 Konklusjon

Arbeidet er fortsatt veldig lærerikt og teamet er meget motivert. Vi har møtt noen utfordringer undervegs, men har lært av de og kommet videre fra disse. Teamet jobber offensivt og har stor tro på både utviklingen og oppgaven.

I.3 Statusrapport 27.04.18

I.3.1 Fremdrift

Løsningen er nå ansett som ferdigstilt. Vi og oppdragsgiver er fornøyd med produktet slik det endte opp og føler at det etterlever kravene med utgangspunkt i de avgrensninger som ble satt. Under de siste par sprintene har vi klart å holde god progresjon på å ferdigstille løsningen, til tross for at mye av ressursene ble satt over til å jobbe med rapporten slik det initielt var planlagt.

Rapporten har per nå fått en god struktur, og vi føler at vi har kontroll på hva som må gjøres og forbedres videre for å ende opp med et godt sluttprodukt også på denne delen av bacheloroppgaven.

I.3.2 Avvik

Denne perioden startet med noen hinder, da det viste seg at vår distributør av overvåkningsdata hadde foretatt noen endringer angående autentisering mot API-et. Vi mistenker at dette kommer av at vi avdekket et sikkerhetshull i deres system, da vi kunne autentisere oss for alle kundekontoer med samme client-id og client-secret. Dette ble heldigvis oppklart relativt tidlig, men det var allikevel et lite tilbakeslag for arbeidsprosessen.

Videre så ble det noe avgrensning i funksjonalitet under denne perioden, da tiden ikke strakk til for å implementere disse. Etter samtale med oppdragsgiver ble det gjort klart at denne tiltenkte funksjonaliteten ikke var av absolutt nødvendighet, og at de fint kunne bruke systemet til sine behov slik det er per i dag.

I.3.3 Konklusjon

Vi er særdeles fornøyd med at vi klarte å utvikle et produkt som oppdragsgiver er fornøyd med og som kan tas i bruk så snart det er ønskelig, til tross for enkelte «motbakker» vi har støtt på undervegs. Nå er det bare å bruke den siste tiden til å skrive ferdig en eksemplarisk rapport som beskriver det ferdige produktet og prosjektets gang i sin helhet.