Vurdering av Felles Studentsystem (FS)

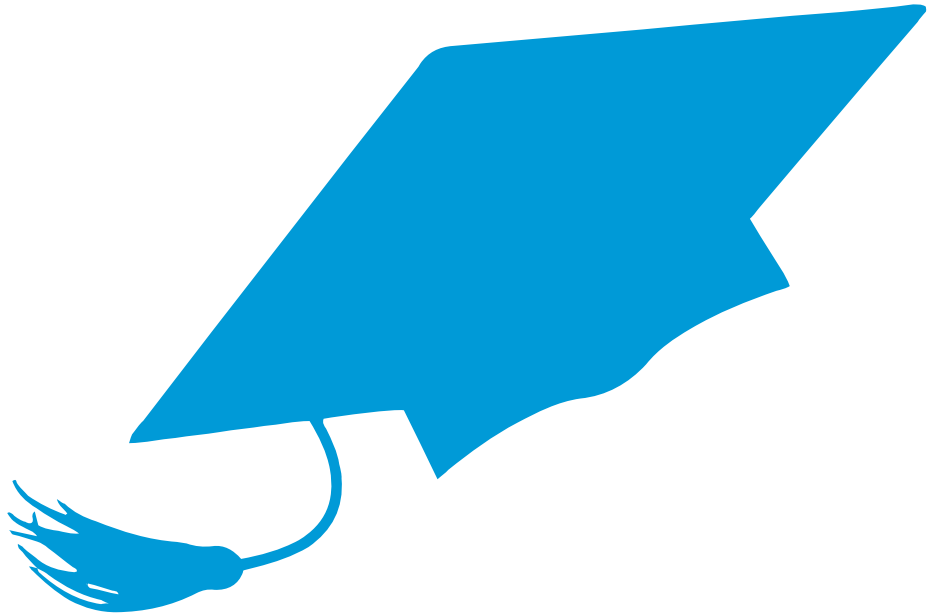
Sluttrapport

Forberedt for: Unit

6. mai 2020 | Versjon #1

Gartner®

Innholdsfortegnelse



Gartner

1. Sammendrag

Sammendrag

Målsetning med vurderingen av Felles Studentsystemet (FS) var tre-delt;

1. Gjennomføre en funksjonell gjennomgang av det studieadministrative området med formål; 1) bruker-/kundetilfredshet av dagens funksjonalitet og 2) fremtidige funksjonelle ønsker og behov
2. Beskrive dagens situasjon og gjennomføre teknisk vurdering av løsningsarkitekturen på applikasjoner, databaser, APIer, mellomvare-/integrasjonsplattformer. Med anbefaling om fremtidig API felles arkitektur og policy.
3. Etablere et treårig veikart med tiltak for modernisering av FS, med grovestimat på gjennomføringskostnad tilknyttet tiltakene

Konklusjonen fra den funksjonelle og tekniske vurderingen er at FS er i teknisk god stand og er et godt utgangspunkt for videre modernisering, og krever ikke en total utskiftning;

- **Funn fra den funksjonelle vurderingen** viser at selv om behovene for det studieadministrative området er dekket godt nok i dag, må det videreutvikles for å bl.a. bli mer brukervennlig, mindre kompleks og mer tilrettelagt for fremtidige behov, arbeidsmåter og trender. Sett fra et IT-perspektiv er det behov for et moderne, fremtidsrettet og forvaltningsbart system med en arkitektur som gjør det mulig å bygge gode brukerorienterte tjenester.
- **Vurderingen fra den tekniske analysen** tilsier at FS bør gjennomgå en kontinuerlig modernisering mot en målarkitektur. FS er i en arkitekturmessig livssyklus, et sted mellom «topp» og «begynnelse av foreldelse» derfor er tidspunktet riktig for modernisering. Ingen områder er vurdert som "røde", dvs. ikke behov for umiddelbar utskifting.

Ny målarkitektur (inkl. API) er utformet etter et sett med prinsipper som medfører at videreutviklingen av applikasjoner vil bli utviklet med mindre tjenester, bygget rundt brukeren («mesh-applikasjon») og integrert ved hjelp av APIer og hendelser.

Andre betraktninger og kritiske suksessfaktorer for å lykkes med videreutvikling av FS er; (i) enighet om overordnede prinsipper for prioritering, (ii) implementering av arkitekturstyring, (iii) forbedring av produkt-/prosjektporteføljestyring for å prioritere rekkefølgen av investeringer effektivt og (iv) nødvendig ressurskapasitet og kompetanse.

Veikart for videre modernisering av FS mot målarkitekturen består av to faser og fire arbeidsstrømmer;

- Fase 1 er forberedelser til gjennomføring av moderniseringen av FS der Unit må sikre nødvendige forutsetninger innenfor bl.a. samhandling, arkitekturstyring og kompetanse (må estimeres nærmere, men grov antagelse er 2-3 FTE i opptil 6 måneder)
- Fase 2 er gjennomføring av moderniseringen og har tre arbeidsstrømmer; 1. Gjøre applikasjoner mer brukersentriske, 2. Klargjøre FS med neste generasjons APIer og 3. Modernisering av FS (inkl. avvikling av PowerBuilder)
- Gjennomføring av FS moderniseringen (Fase 2) er grovestimert til å kreve 5-6 FTE i tillegg til dagens bemanning over 3.5 år

2. Sluttrapport

2.1 Introduksjon

Introduksjon til prosjektet «Vurdering av FS»

Bakgrunn for Vurdering av FS prosjektet

- Felles Studentsystem (FS) er kjernesystem for flere av tjenestene som Unit leverer for utdanningssektoren. Den er delvis utviklet som en plattform som utveksler data og sammenstiller informasjon for ulike brukergrupper.
- Målet med fremtidige FS-tjenester er å levere brukervennlige basisløsninger for studieadministrative oppgaver, masterdata innen studieadministrative data samt tilrettelegge for gjenbruk av data gjennom API og integrasjoner. Dette skal muliggjøre innovasjon og tjenesteutvikling både hos Unit, hos universiteter og høyskoler og av markedsaktører.
- Unit ønsker å gjennomføre en ekstern gjennomgang av FS fra et funksjonelt og teknisk perspektiv.

Målsetning med Vurdering av FS prosjektet

1. Gjennomføre en funksjonell gjennomgang av det studieadministrative området med formål; 1) Bruker-/kundetilfredshet av dagens funksjonalitet og 2) fremtidige funksjonelle ønsker og behov
2. Beskrive dagens situasjon og gjennomføre teknisk vurdering av løsningsarkitekturen på applikasjoner, databaser, APIer, mellomvare-/integrasjonsplattformer. Med anbefaling om fremtidig API felles arkitektur og policy.
3. Etablere et treårig veikart med tiltak for modernisering av FS, med grovestimat på gjennomføringskostnad tilknyttet tiltakene.

Avgrensinger

- Prosjektet vil også se på omfang av kildekode og overordnet teknisk oppsett av komponenter i løsningsarkitekturen, men ikke utføre detaljerte analyser av kildekode og teknisk oppsett/konfigurasjon.

Tilnærming til prosjektet

Januar - Mai

Arbeidet startet opp 29. januar og med sluttpresentasjon og overlevering av sluttrapport til Unit 7. mai

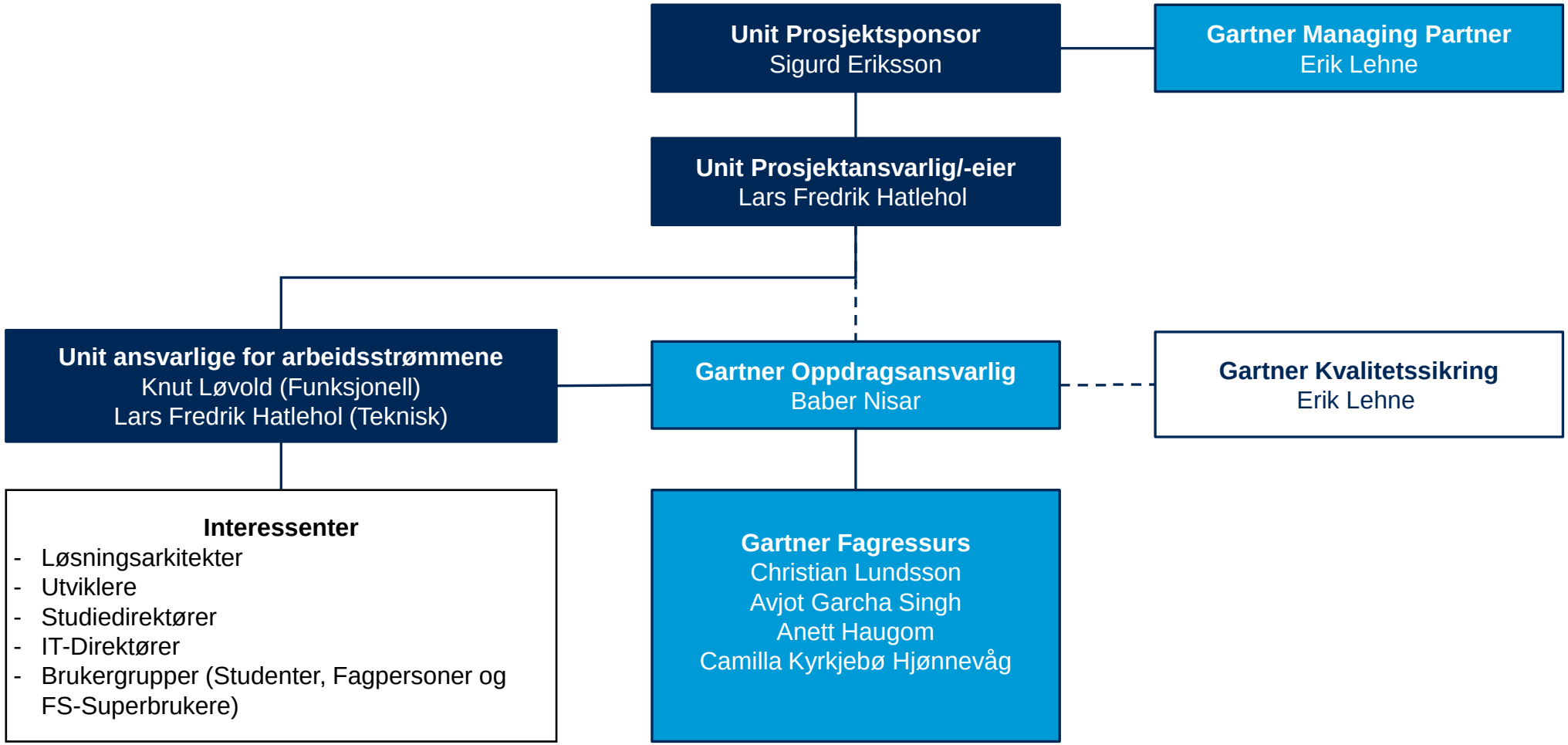
Mai – Juni

Prosjektmobilisering og -initiering	Datainnsamling og analyse av FS	Utlede tiltak og veikart	Ferdigstille sluttrapport og sluttpresentasjon	Presentere sluttrapporten i sektorfora
Mobilisere prosjekt og gjennomføre oppstartsmøte (kick-off) ▪ Gjennomgang av omfang og prosjektplan ▪ Avtale arbeidsfordeling mellom Unit og Gartner ▪ Avstemme målsetning, omfang og tilnærming for innhenting av data	**Datainnsamling** ▪ Gjennomgang av dokumenter relatert til FS ▪ Intervju og nettundersøkelse med interessenter **Analyse** ▪ Gjennomføre analyse og vurdering av FS ▪ Validere funn og vurdering	**Utlede tiltak og veikart** ▪ Utarbeide tiltak ▪ Utarbeide en anbefaling på API fellesarkitektur og policy ▪ Lage et grovestimat på gjennomføringskostnad av tiltak ▪ Validere tiltak, målarkitektur, veikart og estimater	**Ferdigstille sluttrapport** ▪ Utarbeide utkast av sluttrapport ▪ Få tilbakemelding på utkast av sluttrapporten ▪ Ferdigstille sluttrapport **Overlevering av sluttrapport** ▪ Presentere og overlevere sluttrapport til ledelsen i Unit	**Presentere sluttrapport i sektorfora**

Følgende orienteringer av preliminær arbeid ble gjennomført i fagutvalgene

- Fagutvalg for utdanning (1. April) – Presentasjon av preliminær oppsummering fra intervjuene med studiedirektører
- Fagutvalg for IMD (2. April) – Orientering fra prosjektet med presentasjon av preliminære oppsummeringer, funn og konklusjoner

Organisering av prosjektarbeidet



2.2 Funksjonell vurdering

Tilnærming til funksjonell vurdering

Formålet med gjennomføring av den funksjonelle vurderingen av det studieadministrative området var å innhente;

1. bruker-/kundetilfredshet av dagens funksjonalitet
2. fremtidige funksjonelle ønsker og behov

Fem interessentgrupper deltok i den funksjonelle vurderingen

Fokus på strategisk perspektiv		Fokus på sluttbruker perspektiv		
Studiedirektører	**IT-direktører**	**FS-/superbrukere**	**Fagpersoner**	**Studenter**
Studiedirektører fra 13 institusjoner deltok i intervju *Merknad; I intervjuene var det ikke klar skille mellom FS som helhet og FS klienten spesifikt, som Unit har.*	IT direktører fra 5 institusjoner deltok i intervju	86 av 94 (91%) respondenter besvarte nettbaserte spørreundersøkelsen	30 av 38 (79%) respondenter besvarte nettbaserte spørreundersøkelsen	30 av 38 (79%) respondenter besvarte nettbaserte spørreundersøkelsen

Merk at når interessenter blir bedt om å komme med ønsker og behov, så vil det de er godt fornøyd med komme i bakgrunnen.

Oppsummering fra intervjuene med studiedirektører

1. **FS dekker mer enn godt grunnleggende studieadministrative behov og funksjoner i dag**, men måten funksjonene tas i bruk gjenspeiler ikke hvordan et moderne universitet/høgskole jobber i dag. Dette medfører stor grad av manuelt arbeid og omstendelige arbeidsoppgaver. Det er ytret usikkerhet knyttet til hvor godt FS skal klare å dekke komplekse behov/arbeidsflyt og det er gitt uttrykk for at videreutvikling må prioriteres.

2. FS bærer preg av en pre-digital hverdag med mye manuell kontroll og papir. Det er gitt eksempler der **lav brukervennlighet** og **komplekst brukergrensesnitt** har medført kritiske feil med konsekvenser for institusjonene og for studentene. Tungvinte arbeidsoppgaver og saksbehandling fører også til mye dobbelt arbeid og en ustrukturert arbeidsflyt.

3. Det er utfordringer som ikke er løst og muligheter som ikke er dekket av dagens plattform. Ønsker for **fremtiden inkl. bedre samspill og deling mellom institusjoner, mer selvbetjening og automatisering, mer personlige saksbehandling, samt bedre støtte for internasjonalisering, livslang læring og læringsanalyse.**

4. **Suksesskriterier for å bedre understøtte moderne arbeidsprosesser** er økt samhandling med deling på tvers av institusjonsgrenser, et raskere utviklingstempo, forenklet brukergrensesnitt, mer sømløst dataflyt, forbedre dataintegritet, bedre utnyttelse av sektorkompetanse, forutsigbar finansiering, m.m.

5. Studiedirektørene er svært opptatt av at sektoren opparbeider seg et enhetlig syn over alle studentene i Norge og mener at FS er kritisk for å få dette til. De fleste ser for seg **en fremtid der FS er en kjerne for nasjonal studentdata**, en autoritativ kilde som **muliggjør deling på tvers av institusjonsgrenser.**

6. **Ved videreutvikling av FS antar studiedirektørene at det vil kunne utløse betydelige gevinster når det gjelder effektivitet, kvalitet og strategisk måloppnåelse**. Det forventes at gevinstene vil bidra til effekter som f.eks. økt produktivitet av studieadministrative ansatte, økt kvalitet i studieløpene, økt fellesskapsfølelse på sektor nivå, m.m.

Se seksjon 3.4 - s.118-126 for detaljer

Gartner

Oppsummering fra intervjuene med IT-direktører

1. Mange lokale systemer hos institusjonene har en avhengighet til FS. Det gis uttrykk for at **det ikke er teknisk og arkitekturmessig enkelt å forholde seg til FS**. Den mangler et godt integrasjonslag, har mangelfulle APIer og arkitekturen oppfattes litt kompleks og lite helhetlig. Det nevnes også utfordringer med masterdata fra FS pga. ulik definisjonsbruk, som gjør integrasjonsjobben vanskelig og å sammenstille informasjon som er relevant for alle.

2. Det integreres veldig nær/mot databasen, og ikke gjennom en veldefinert n-lags arkitektur. Dette medfører til at man mister sikkerhet og annen logikk i mellom, er tungvint å jobbe med, må lage skreddersydde løsninger (som bl.a. gjør at det tar lengre tid å levere, ikke alt kan gjøres og integrasjoner kan knekke), som fører til at **FS blir ikke en god plattform for å gjøre utvikling på**. Ikke alle institusjoner i sektoren har lokal utviklingskompetanse for å gjøre integrasjonsarbeid mot FS, noen bruker eksterne aktører, og for mange mindre vil dette ha konsekvenser for nyutvikling av lokale tjenester.

3. Fremtidig behov og ønsker for FS (i en plattform kontekst) er et **moderne og fremtidsrettet system og arkitektur**, som er brukerorientert og forvaltningsbar slik at flere gode tjenester kan bygges. Et plattform som har **standardiserte grensesnitt, mikrotjenester og APIer som er sikre, robuste, skalerbare og enklere å konsumere** som gir en plattform med;
   - Enklere datautveksling med mulighet for å skrive tilbake til FS og sikre datatilgang i henhold til lover og forskrifter (f.eks. GDPR regelverk)
   - Bedre støtte for håndtering av internasjonale brukere (f.eks. med brukerautentisering og i datalagring) og mobilitet (på tvers av campus miljøer)
   - Bedre støtte for prosessautomatisering og mulighet for at ulike brukermiljøer kan hente ut data uten behov for involvering av IT
   - Større tilgang til data som ligger klare til bruk med standard grensesnitt og åpne APIer for f.eks. læringsanalyse
   - Fleksibilitet for enklere og raskere utvikling av nye lokale behov hos institusjonene for f.eks. smart campus
   - Tilrettelagt for enklere og raskere integrasjon med andre moduler fra markedet
   - Tilpasningsmulighet til fremtidige trender og nye tjenester/løsninger innenfor utdanning og læring i og utenfor Norge

4. For å **muliggjøre at en slik plattform blir en suksess** er det viktig å definere hva er kjernen i FS og hvordan rendyrke den, samt hvordan FS inngår på tvers av nasjonale løsninger og hva skal dekkes av andre løsninger. Ha på plass en godt definert informasjonsmodell, veldefinert n-lags arkitektur, modularisere FS for enklere håndtering og videreutvikling ved å tenke tilnærming med f.eks. mikrotjenester. FS med studentdata må også passe inn helhetlig med pågående IAM sektorprosjektet. God forvaltning og sikkerhet rundt data er viktig for å få konsistente data som er en vesentlig faktor for bruk.

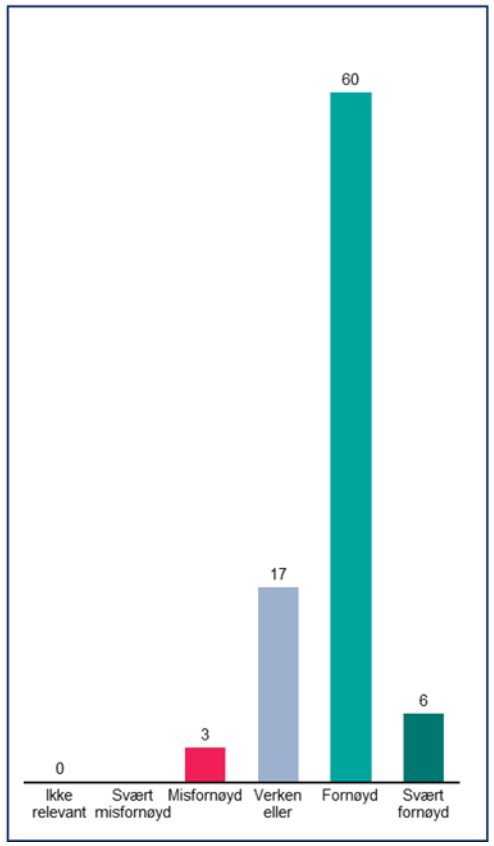
5. **Viktige forutsetninger og suksesskriterier** for videreutvikling av FS som nevnes er; forutsigbar finansiering, nødvending kompetanse/kapasitet, raskere tempo i utviklingen (dagens tempo er for sakte), forbedring av styrings- og forvaltningsmodellen med bl.a. tydeligere kommunikasjon åpenhet rundt styring og prioritering, og bedre involvering av sektoren med forankring av arkitekturen hos institusjonene også.

Oppsummering fra brukertilfredshet brukergrupper FS-/superbruker, fagperson og student

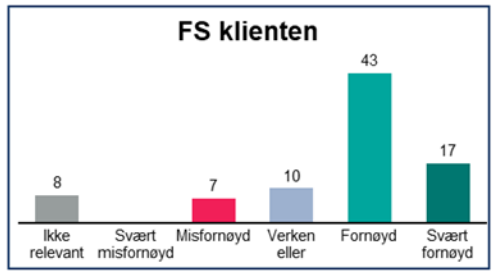
FS-/superbruker (86 respondenter)

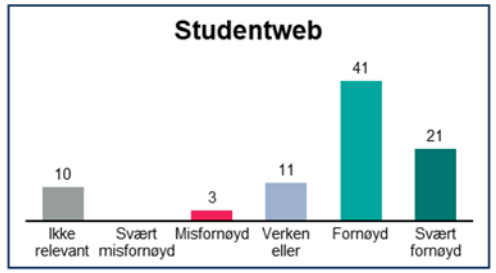
Hvor fornøyd* er du med…

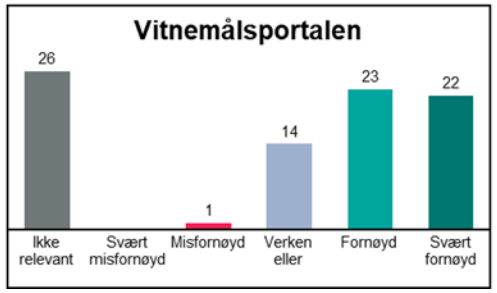
FS som helhet



Topp 3 mest relevante applikasjoner

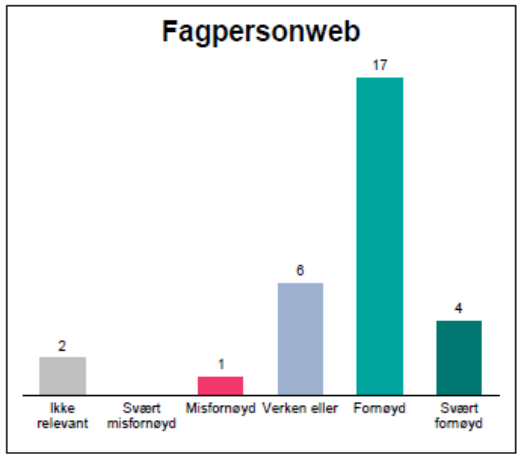


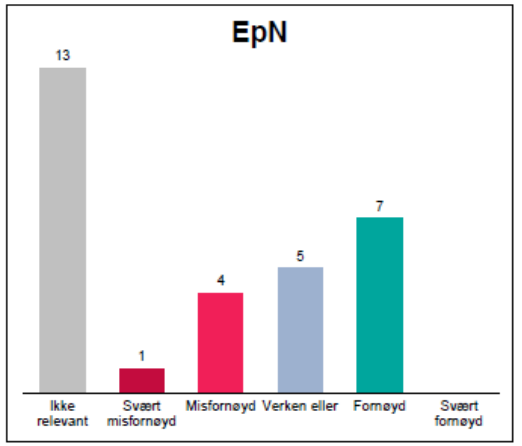




Fagperson (30 respondenter)

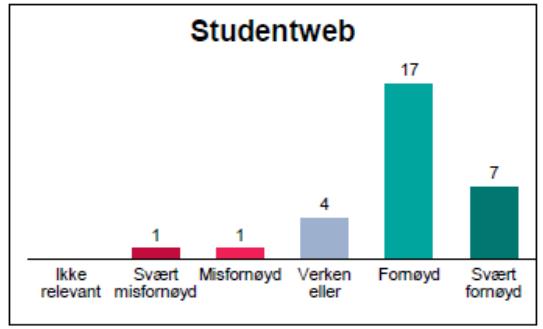
Hvor fornøyd* er du med…

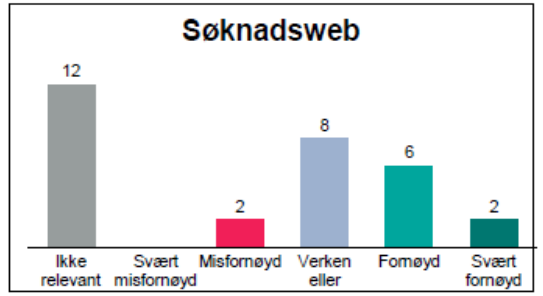


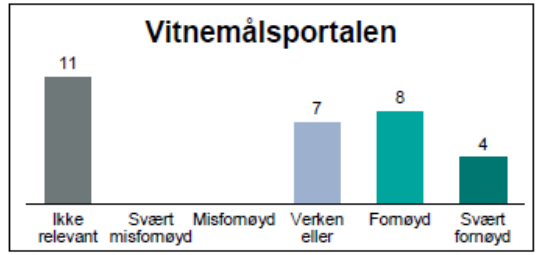


Student (30 respondenter)

Hvor fornøyd* er du med…







Se seksjon 3.4 - s.128-149 for utdrag og vedlagte filer med detaljer fra brukerundersøkelsen

Gartner

Oppsummering fra forbedringer brukergrupper FS-/superbruker, fagperson og student

FS-/superbruker (86 respondenter)	Fagperson (30 respondenter)	Student (30 respondenter)
348 tilbakemeldinger fordelt etter applikasjon • 103 FS i sin helhet, • 99 FS klienten • 45 EpN • 101 Andre	47 tilbakemeldinger fordelt etter applikasjon • 12 EpN • 35 Fagpersonweb	43 tilbakemeldinger fordelt etter applikasjon • 35 Studentweb • 7 Søknadsweb • 1 Vitnemålsportalen

Nøkkelfunn fra forbedringer
• **Flest forslag i følgende applikasjoner knyttet til:**
  • Brukervennlighet i FS klienten
  • Bedre prosesstøtte / funksjonalitet og brukervennlighet i EpN
  • Forbedring av brukervennlighet i Studentweb
• **Flest forslag går ut på forbedring av brukervennlighet**
  • Mer intuitiv navigasjon
  • Mer forståelige feilmeldinger
  • Flere hjelpetekster.
  • Mer effektiv arbeidsutførelse gjennom prosessoptimalisering, integrasjoner, automatiseringer
• **Mer funksjonalitet for bedre prosesstøtte:**
  • Enkelte oppgaver kan ikke utføres optimalt i FS pga manglende funksjonalitet. Mange forslag går derfor ut på legge til funksjonalitet (småjusteringer).
• **Mer tilgang til data/informasjon**
  • Flere/bedre/mer tilgjengelige rapporter
  • Mer integrerte systemer
• **Bedre brukergrensesnitt**
  • Mer moderne og intuitiv design – oppdatere farger, symboler, figurer, skriftstørrelse
  • Brukergrensesnitt tilpasset mobil og flere nettlesere
	Nøkkelfunn fra forbedringer
	• **Fagpersonweb har flest forslag til forbedringer:**
	  • Behov for integrasjoner
	  • Behov for data/informasjon
	  • Bedre brukervennlighet
	• **Mange forbedringer i EpN knyttet til studieelementer**
	  • Bedre brukervennlighet
	• **Behov for integrasjoner til andre systemer i Fagpersonweb**
	  • For å kunne utføre oppgaver effektivt i Fagpersonweb, spesielt LMS systemer (Canvas) og sensursystemer (Inspera, Wiseflow, MittUiB)
	• **Færre systemer for økt brukervennlighet**
	  • Faglærere ønsker færre systemer å forholde seg til knyttet til studieadministrasjon
	  • Utførelse av samme type oppgaver i flere systemer medfører en rekke problemer
		Nøkkelfunn fra forbedringer
		• **Flest ønsker forbedringer i Studentweb, men denne er også mest brukt**
		• **Forbedringene gjelder primært brukervennlighet og brukergrensesnitt**
		  • Relevant informasjon på forsiden
		  • Mer veiledende tekster og datakontroller
		  • Mer intuitiv symbol-, farge- og tekstbruk
		  • Opplæring i funksjonalitet
		  • Brukergrensesnitt tilpasset mobil
		• **Flest forbedringer i Studentweb knyttes til**
		  • Studentweb som helhet
		  • Utdanningsplan
		  • Informasjon om rettigheter og plikter
		  • Begrunnelse og klage
		• **Forbedringer i Søknadsweb**
		  • Opplasting av dokumentasjon ifm søknad
		  • Søknadsweb i sin helhet

Se seksjon 3.4 - s.128-149 for utdrag og vedlagte filer med detaljer fra brukerundersøkelsen

I tillegg til behov ytret av interessentene er det flere trender Gartner identifiserer innenfor i UH-sektor som kan være relevante

Bygge opplevelser	Skape tillit
Smart Campus	Reinventing Credentials
Collegiate Esports	Ethical Use of Data
Nudge Tech	Reframing Security
New Display Tech	Digital Credentials
Career Software	Faculty Info Systems

Forenkle hverdagen	Drive transformasjon
Analytics Everywhere	New Business Models
5G Infrastructure	Artificial Intelligence
Online Differentiation	Creative Financing
Cross-Lifecycle CRM	Corporate Collaboration
Digital Dexterity	Ecosystem

Forklaring: Business trends (uthevet i blå) og Strategic technology trends (i svart)

Source: *Top 10 Business Trends Impacting Higher Education in 2020* (Gartner Research ID: G00465204)

Gartners analyse av forretningstrender og strategisk teknologi for høyere utdanning gir et utgangspunkt for fire strategiske utviklingstrekk eller «megatrender» for sektoren.

Disse utviklingstrekkene kan gi Unit en indikasjon på fremtidig retning og ambisjonsnivå for FS. Trendene sett i et strategisk perspektiv kan utfordre tempo og strategisk retning for videreutvikling av FS.

Videreutviklingen iht disse trendene kan medføre:

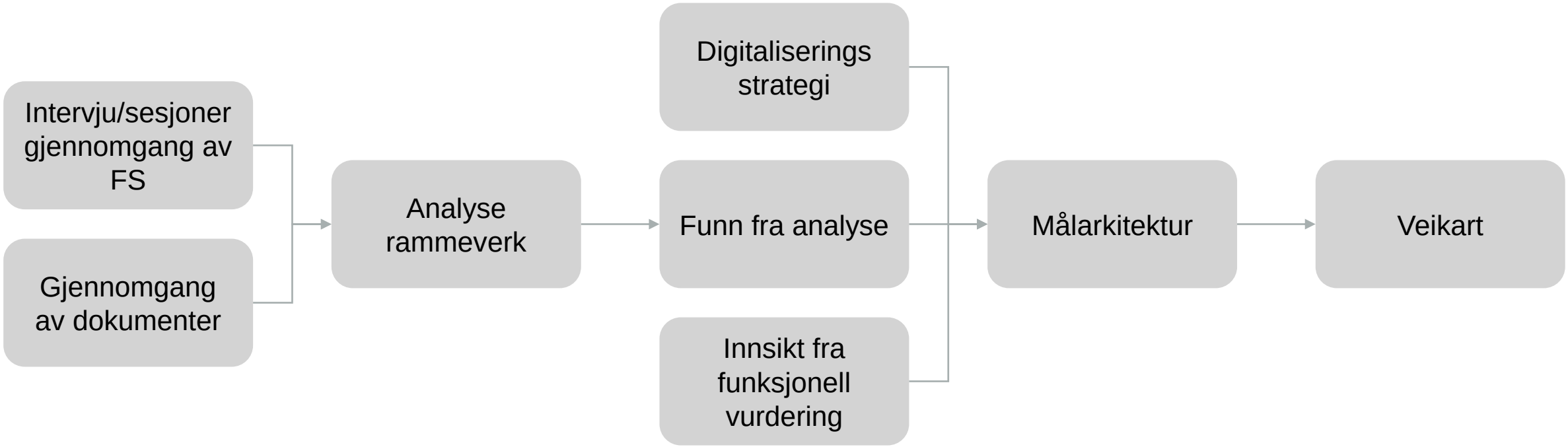
- Gode og bedre brukeropplevelser
- Økt tillit og legitimitet hos studenter, ansatte, andre utdanningsinstitusjoner og næringslivet
- Forenklet hverdag for brukere og studenter
- Innovasjon og drivkraft for transformasjon av tjenester for dagens studenter, morgendagens studenter, vitenskapelige og administrative ansatte

Se i Vedlegg s.152-153 for kort forklaring av trendene.

Gartner

2.3 Teknisk vurdering

Tilnærming til analysen og utarbeidelse av API målarkitektur og veikartet



10+ intervju og sesjoner med utviklere og arkitekter fra Unit og Uninett

ISO 25010 rammeverk for SW Quality og Gartner tilnærming til software assessments

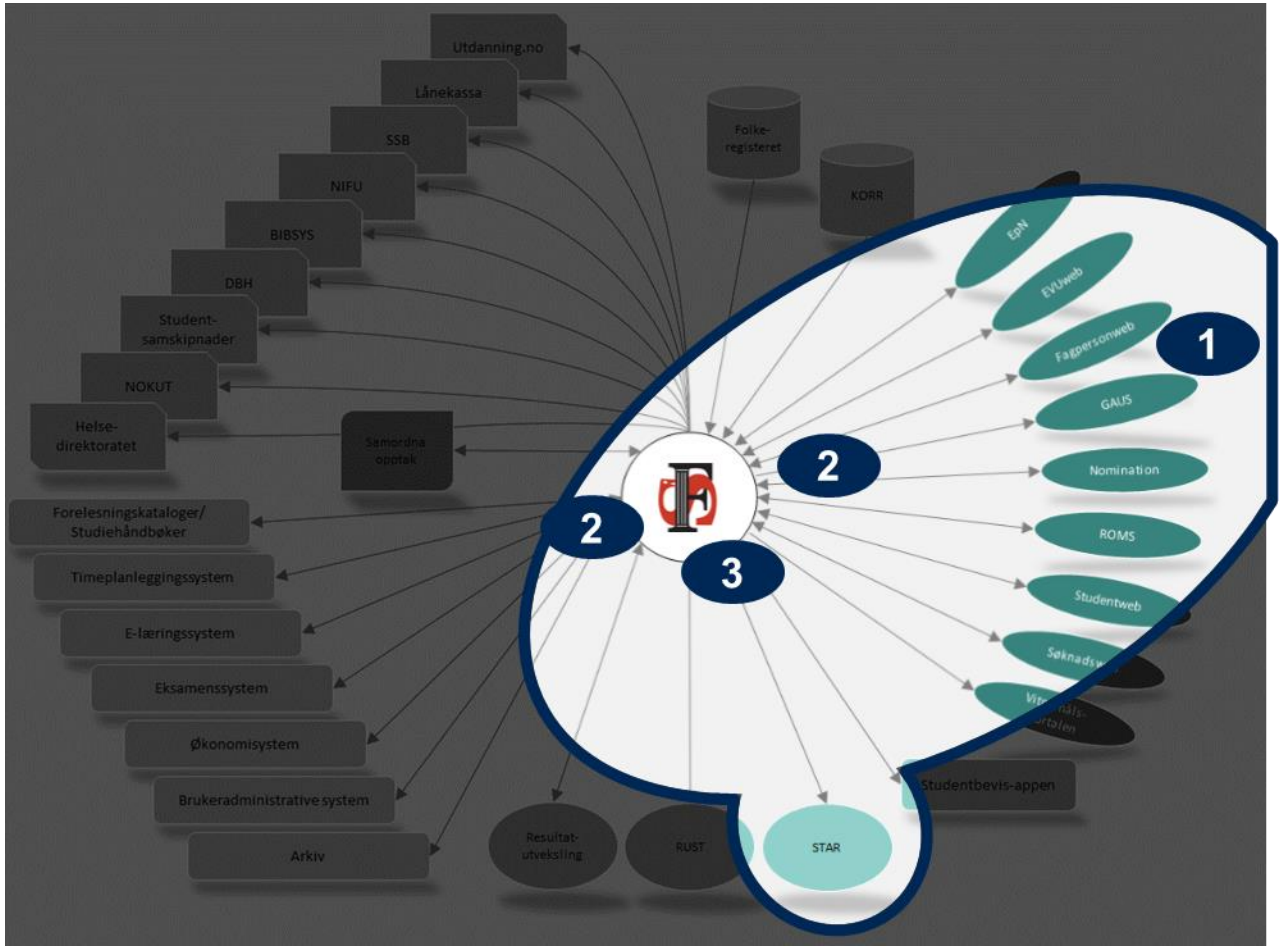
Innsikt fra 3 brukerundersøkelser (student, fagperson og FS-/superbruker) og intervju med studiedirektører og IT-direktører

Sikre at prinsipper og krav er oppfylt

Rekkefølge av tiltak som må gjøres for transformasjon

Arkitekturlag som er del av analysen for teknisk vurdering inkluderer FS front-end applikasjoner, integrasjoner og database

I analysen inngår deler av FS som omfatter integrasjoner og front-end applikasjoner rundt FS databasen…



Struktureres som følgende for videre analyser…

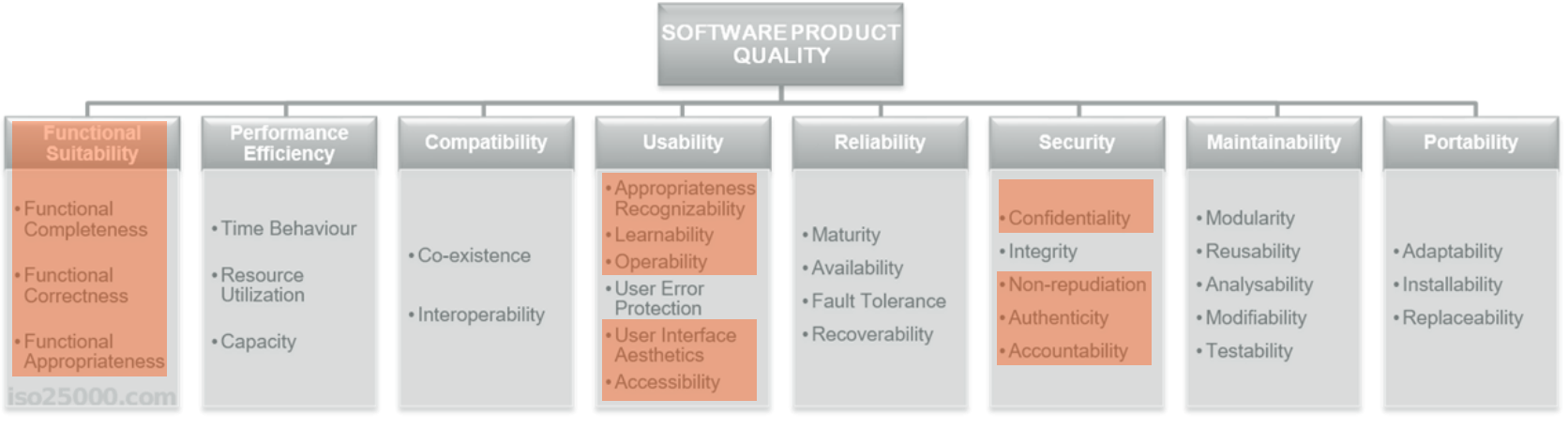
1. FS brukerapplikasjoner (UI)

2. FS integrasjoner og API (API)

3. FS databasen (Back-end)

Gartner

Relevante dimensjoner og kategorier fra ISO 25010 rammeverket for Software Quality ble brukt for å analysere FS



1. FS brukerapplikasjoner (UI)
2. FS integrasjoner og API (API)
3. FS databasen (Back-end)

Forklaring Ikke relevant/out-of-scope i teknisk vurdering

Vurderingen fra analysen tilsier at FS bør gjennomgå en kontinuerlig modernisering mot et målarkitektur

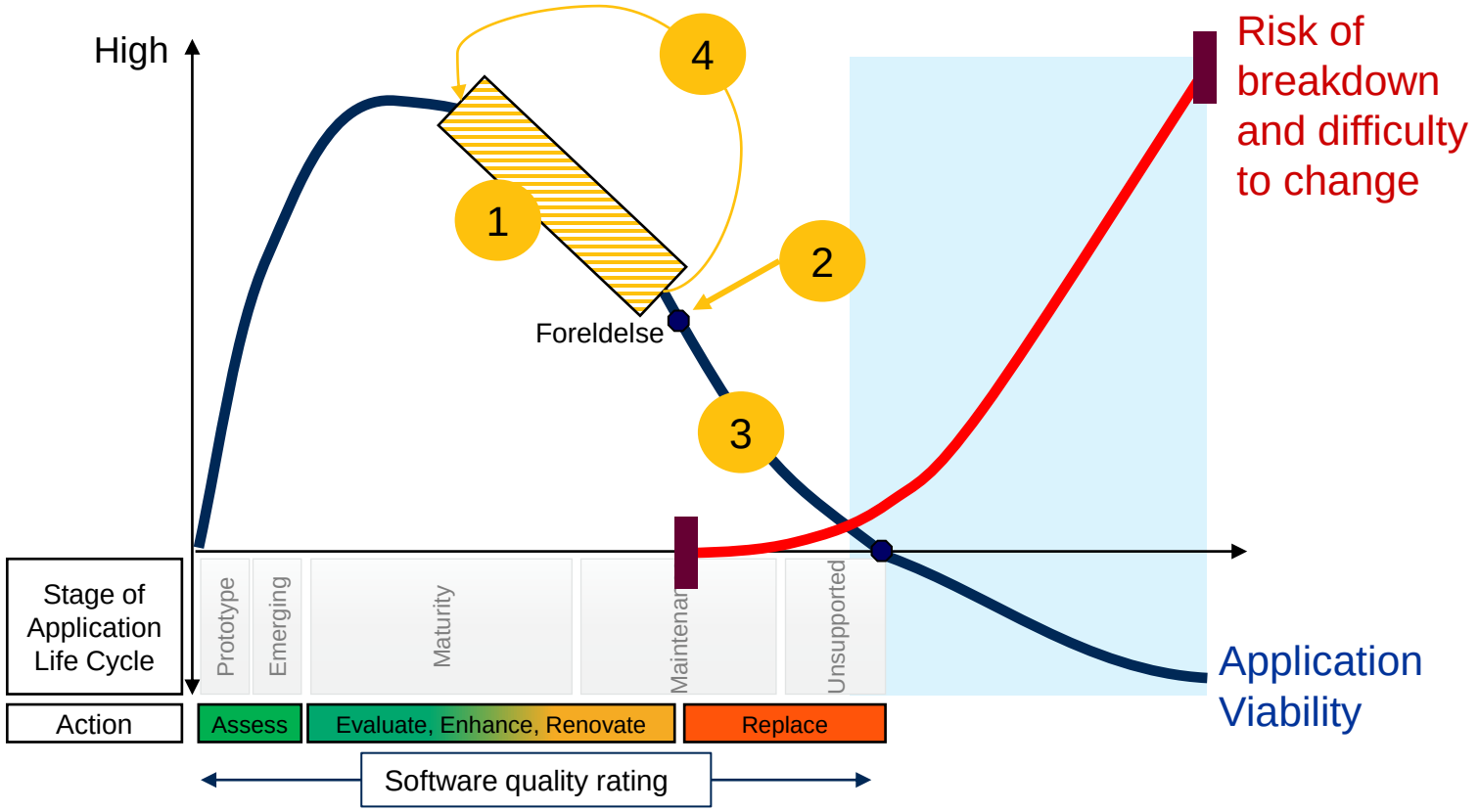
Oversikt over analysen av FS

ISO Category	ISO Dimension	UI	API	Back end	ISO Dimension	UI	API	Back end
Performance efficiency	Time behaviour				Resource utilization			
	Capacity							
Compatability	Co-existence				Inter-operability	N/A		
Usability	Appropriateness	Out of scope			Learnability	Out of scope		
	Operability	Out of scope			End user error protection		N/A	N/A
	User interphase aestetics	Out of scope			Accessibility	Out of scope		
Reliability	Maturity				Availability			
	Fault tolerance				Recoverability			
Security	Confidentiality	Out of scope			Integrity		N/A	N/A
	Non-repudiation	Out of scope			Accountability	Out of scope		
	Authenticity	Out of scope						
Maintainability	Modularity				Reusability			
	Analysability				Modifiability			
	Testability							
Portability	Adaptability				Installability			
	Replaceability							

- Vurderingen av FS er i tråd med hva vi forventer at en skreddersydd løsning vil se ut etter 20+ års utvikling
- Det er **områder som er bygget på moderne teknologier** og i tråd med god praksis
- **Ingen områder er vurdert som "røde",** en vurdering av alvorlig kritikk der tilstanden er dårlig og vanligvis trenger utskifting
- **Det er flere områder som er klassifisert som "gule"**, noe som indikerer at det ikke er optimalt eller i tråd med beste praksis, men det er mulig å utføre rearchitect/refactor til en moderne og optimal struktur

Se seksjon 3.2 - s.40-97 for detaljanalysen med oversikt over funn og forslag til tiltak

FS er i en arkitekturmessig livssyklus et sted mellom «topp» og «begynnelse av foreldelse» derfor er tidspunktet riktig for modernisering



Konklusjoner fra den tekniske vurderingen:

1. Tilstanden spenner fra ganske bra («topp») tilstand til å nærme seg et vippepunkt for foreldelse
2. Nå er det en godt tidspunkt for å oppdatere/ modernisere delene som nærmer seg vippepunktet for foreldelse
3. Når du er for langt ned på kurven for «Application Viability», synker kvaliteten og det blir langt mer komplekst å endre
4. For å unngå foreldelse, må løsningen undersøkes og moderniseres for å gjøre den moderne igjen

Med foreldelse menes start på akselerende verdiforringelse, kostnadene (inklusive vedlikeholdskostnader) øker, og bruksnytten faller.

Gartner

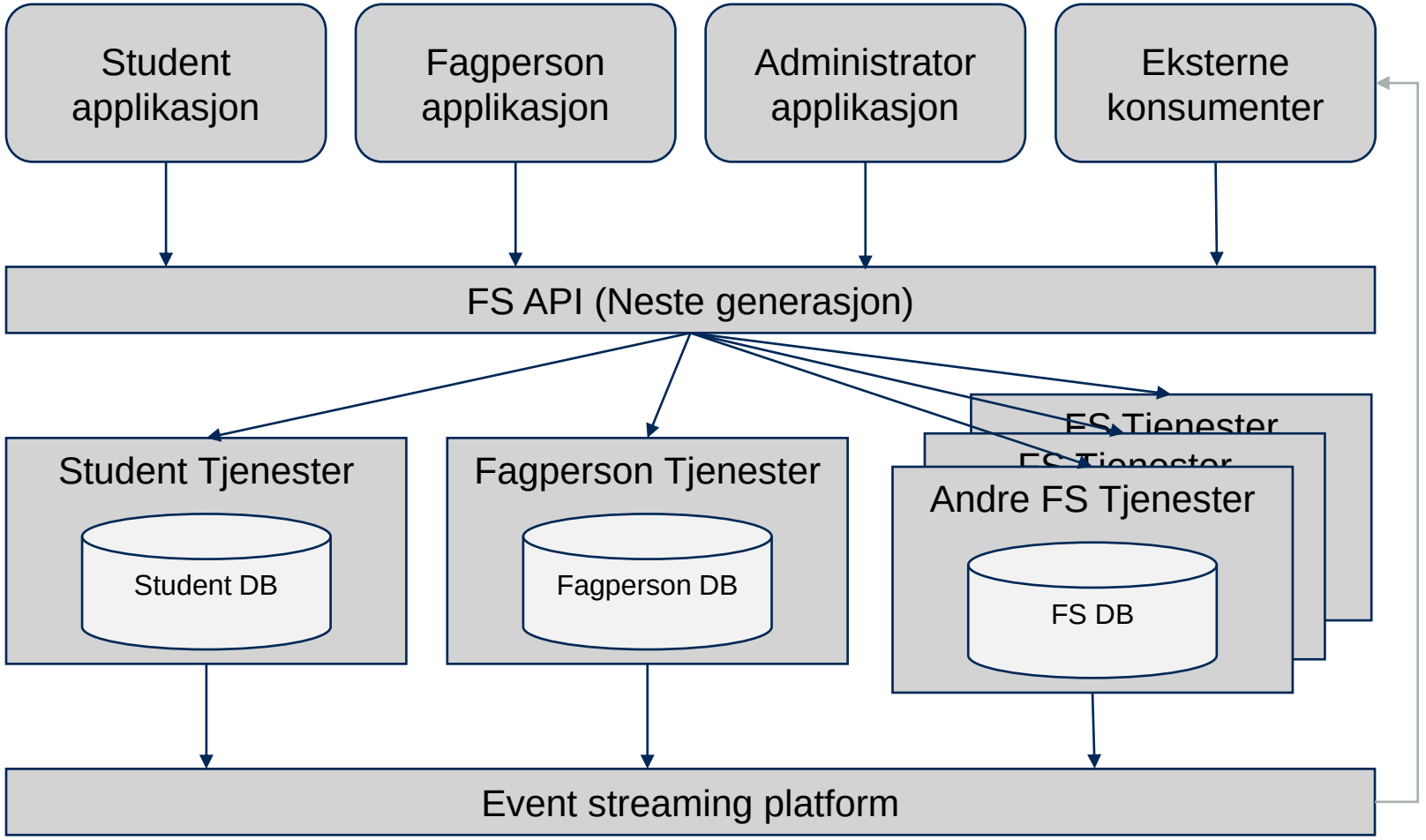
2.4 Målarkitektur

Målarkitektur er utformet etter en sett med prinsipper

	Prinsipp…	Rasjonale…	Har implikasjoner…
1	Nåværende «informasjonseierskap» skal ikke være førende for design av systemet	Data eierskap, fysisk lagring og hvor data konsumeres kan være forskjellige. Fokuset bør være på brukeropplevelse.	Den nåværende tanken om hvem som “eier” data (f.eks. studentdata) må revideres
2	Sluttbrukerløsninger bør være en “mesh” av løsninger basert på en rolle/personas, istedenfor en rekke spesifikke applikasjoner	Vil forbedre brukeropplevelsen, samt lettere å gi et konsistent utseende og preg. Gjør det mulig å optimalisere utviklingsarbeid og infrastruktur.	En rolle har kun en applikasjon
3	Alle dataressurser og use-case handlinger skal være tilgjengelige gjennom bruk av APIer	Vil gjøre bruk av API-er lettere, siden de ikke vil kreve den samme mengden av domenekunnskap.	Det som skal utføres, skal være mulig å utføre med bruk av APIer
4	Ethvert API skal kunne eksponeres for enhver type konsument (ved behov)	Vil muliggjøre raskere utrulling av samme / lignende funksjonalitet på andre plattformer.	APIer utvikles for gjenbruk

I målarkitekturen, er det foreslått å transformere FS APIer i en fremtidsrettet integrasjonsarkitektur. En FS API transformasjon har også implikasjoner for FS arkitekturen, siden FS database må også ivareta noen designkonsepter for å støtte den nye arkitekturen (eksempelvis bruk av mikro-/mini-tjenester).

Konseptuell målarkitektur for FS



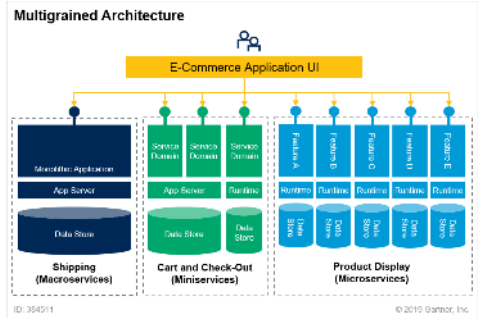
- **Sluttbruker applikasjoner er bygget rundt en personas/rolle**. Dvs. det er ett studentprogram hvor brukeren logger seg på som student, og ikke starter med å velge institusjon
- **Eksterne konsumenter**, kan aksessere alle «FS» tjenester (inkl. data) som er brukt av front-end applikasjoner
- Tilgang til alle backend-tjenester er gjort ved hjelp av neste generasjon **FS API** (inkludert applikasjoner, integrasjon av andre systemer, åpne APIer osv.)
- Back-end blir **delt inn i mindre tjenester,** det vil si deler av back-end og databaser som hører sammen
- En «event streaming platform» sikrer at alle tjenester innenfor «FS-omfang» eller som del av økosystemet blir varslet med relevante endringer

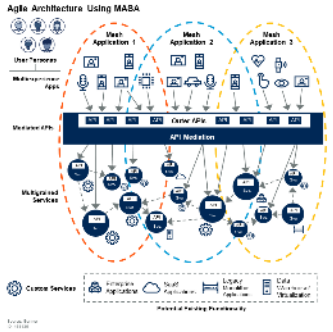
Ny målarkitektur vil ha betydning for videreutvikling av FS

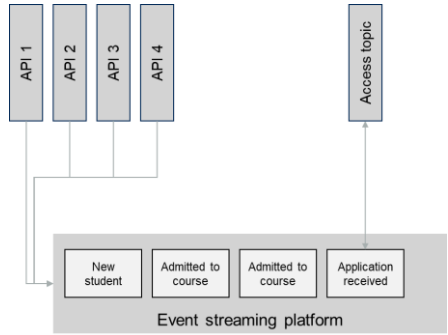
Applikasjoner vil bli utviklet med mindre tjenester	• **For å forbedre den generelle smidigheten** ønsker vi å dele systemet i mindre komponenter med data og funksjonalitet som hører sammen
	• Tjenesten skal organiseres rundt et domene eller sett med data som hører sammen, et av de mest **åpenbare eksemplene er en "student"**
	  - Ved å lage en studentdatabase og et sett med studenttjenester kan vi administrere alle studentrelevante funksjoner på ett sted
	  - Ved å lage denne tjenesten og databasen, kan studentapplikasjonen forbedre sin smidighet og **utvikle relevante funksjonaliteter uten å være helt avhengig av alt annet**

Applikasjoner vil bli bygget som «mesh-applikasjon»	• Mesh-applikasjon betyr at applikasjonen er:
	  - **Bygget rundt en persona, f.eks. en student**
	  - **Kan få tilgang til alle relevante tjenester**
	• En student-app skal være inngangspunktet for studentene med tilgang til eksempelvis:
	  - Søknad
	  - Selvbetjening av data og andre funksjoner
	  - Tilgang til vitnemål, resultater, pågående aktiviteter
	  - Fungere som «kommunikasjons-hub» mellom studenten og de andre relevante delene, f.eks. institusjoner, kursadministratorer, lærere etc.

Applikasjoner vil bli integrert ved hjelp av APIer og hendelser	• **APIer/hendelser skal være den eneste måten å konsumere data eller utføre funksjoner**
	• Det vil sannsynligvis **bli færre API-er**, men mer nyttige siden de vil løse en spesifikk bruksområde **og få tilgang til uten inngående forståelse av FS-datamodellen**
	• En **plattform for strømming av hendelser** vil sikre at forretningsrelevante hendelser blir gjort tilgjengelig for forbrukerne som trenger disse dataene
	• Det er et par viktige endringer sammenlignet med dagens bruk av meldingskøer:
	  - Hendelser tømmes ikke ut av strømmen, en tjeneste kan når som helst "spole tilbake" og begynne å behandle strømmen om nødvendig
	  - Hendelser har ikke en utpekt mottaker, tjenester kan selv styre hvilke hendelser de ønsker å konsumere







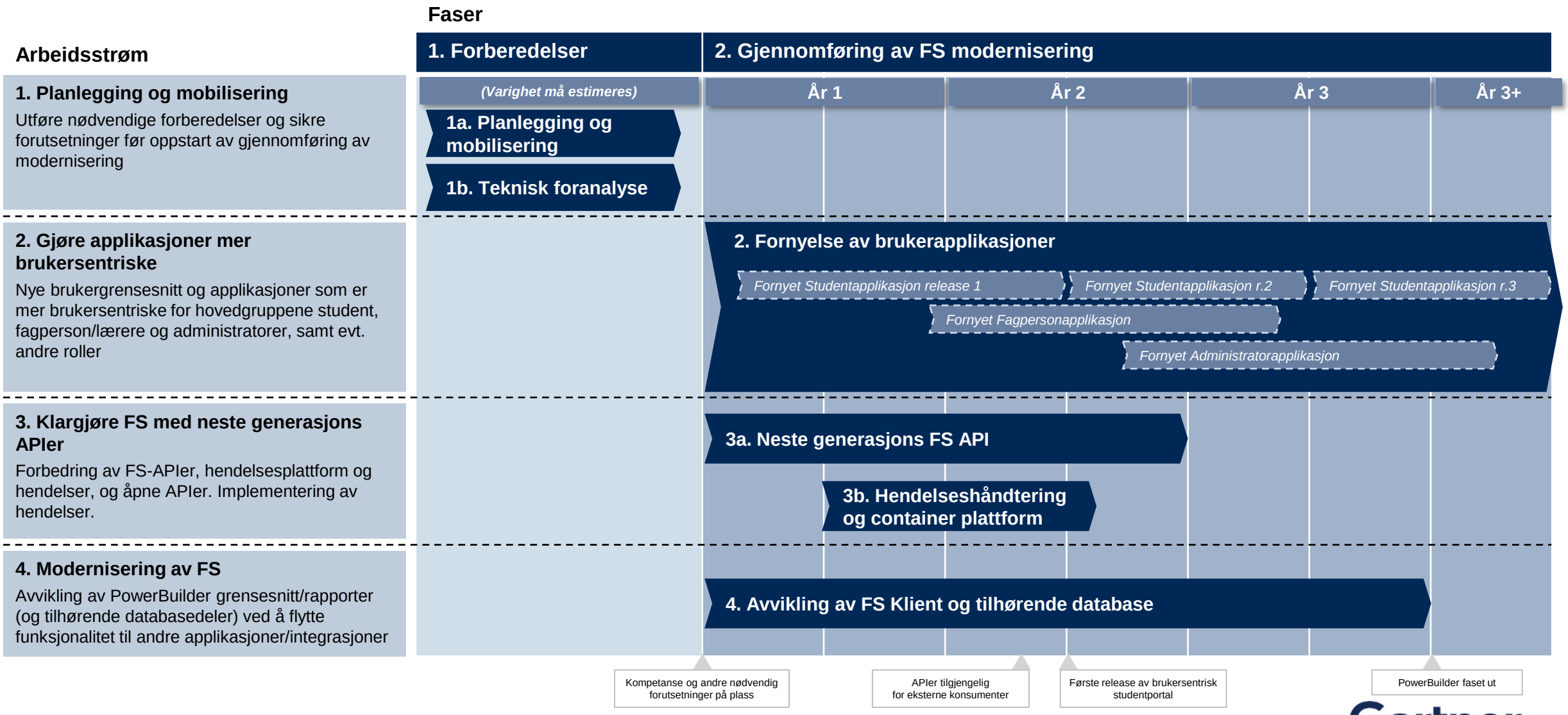
Andre betraktninger og kritiske suksessfaktorer for å lykkes med videreutvikling av FS

Observasjon…	Sikre i videreutvikling…
Lav kost, høy smidighet og høy kvalitet er ikke mulig å kombinere, og noen beslutninger er tatt med antakelsen om å minimere utviklingskostnadene som har skapt noen mindre optimale løsninger	Beslutning på strategisk nivå på hva som er viktig og prioritert
Så langt er det tatt mange avgjørelser uten en god analyse, kobling til en helhetlig arkitektur og teknologivalg og et langsiktig mål	Implementering av arkitekturstyring
En årsak til dagens situasjon er mangelen på god prioritering mellom prosjekter som har ført til overlappende utviklingsinnsats	Forbedring av produkt-/prosjektporteføljestyring for å prioritere rekkefølgen av investeringer effektivt
Behov for å styrke personalets kapasitet og kompetanse for å styre effektiv utvikling med teknologi etter beste praksis	Nødvendig ressurskapasitet og kompetanse

Sikring av disse faktorene i videreutvikling av FS vil bidra til å forbedre ressursproduktiviteten til å utvikle økosystemet av FS applikasjoner. Dette vil bidra til å løse utfordringer som forhindrer ressurser fra å være fullt produktive i dag, slik at bl.a. duplisert arbeid unngås.

2.5 Veikart for videre arbeid

Treårig veikart for videre modernisering av FS mot målarkitekturen



Se seksjon 3.1 - s.32-39 for beskrivelser av veikartsaktiviteter

Forklaring *Forslag til rekkefølge, men bestemmes i fasen før (1. forberedelser)* ▲ Noen hovedmilepæl

Gartner

Gjennomføring av FS moderniseringen er grovestimert til å kreve 5-6 FTE i tillegg til dagens bemanning over 3.5 år

Involvering i arbeidsstrømmene	Eksisterende forvaltningsorganisasjon	1. Arkitektur og governance	2. Front-end utvikling	3. API utvikling
1. Planlegging og mobilisering Utføre nødvendige forberedelser og sikre forutsetninger før oppstart av gjennomføring av modernisering	√	√		
2. Gjøre applikasjoner mer brukersentriske Nye brukergrensesnitt og applikasjoner som er mer brukersentriske for hovedgruppene student, fagperson/lærere og administratorer, samt evt. andre roller	√	√	√	
3. Klargjøre FS med neste generasjons APIer Forbedring av FS-APIer, hendelsesplattform og hendelser, og åpne APIer. Implementering av hendelser.	√	√		√
4. Modernisering av FS Avvikling av PowerBuilder grensesnitt/rapporter (og tilhørende databasedeler) ved å flytte funksjonalitet til andre applikasjoner/integrasjoner	√	√		

Gartner foreslår ressursøkning* med kompetanse innenfor følgende områder

1. Arkitektur og Governance (1 FTE)
- Forvaltning og styring av arkitektur og teknologier i forhold til preferanser og sektor valg/behov
- Utarbeide målarkitektur og designmønster

2. Frontend utvikling (2-3 FTE)
- Webapplikasjonsutviklere for å styrke applikasjonsteamet og begynne å bygge den nye studentapplikasjonen
- Flytter over til Fagperson- og administrator programmene

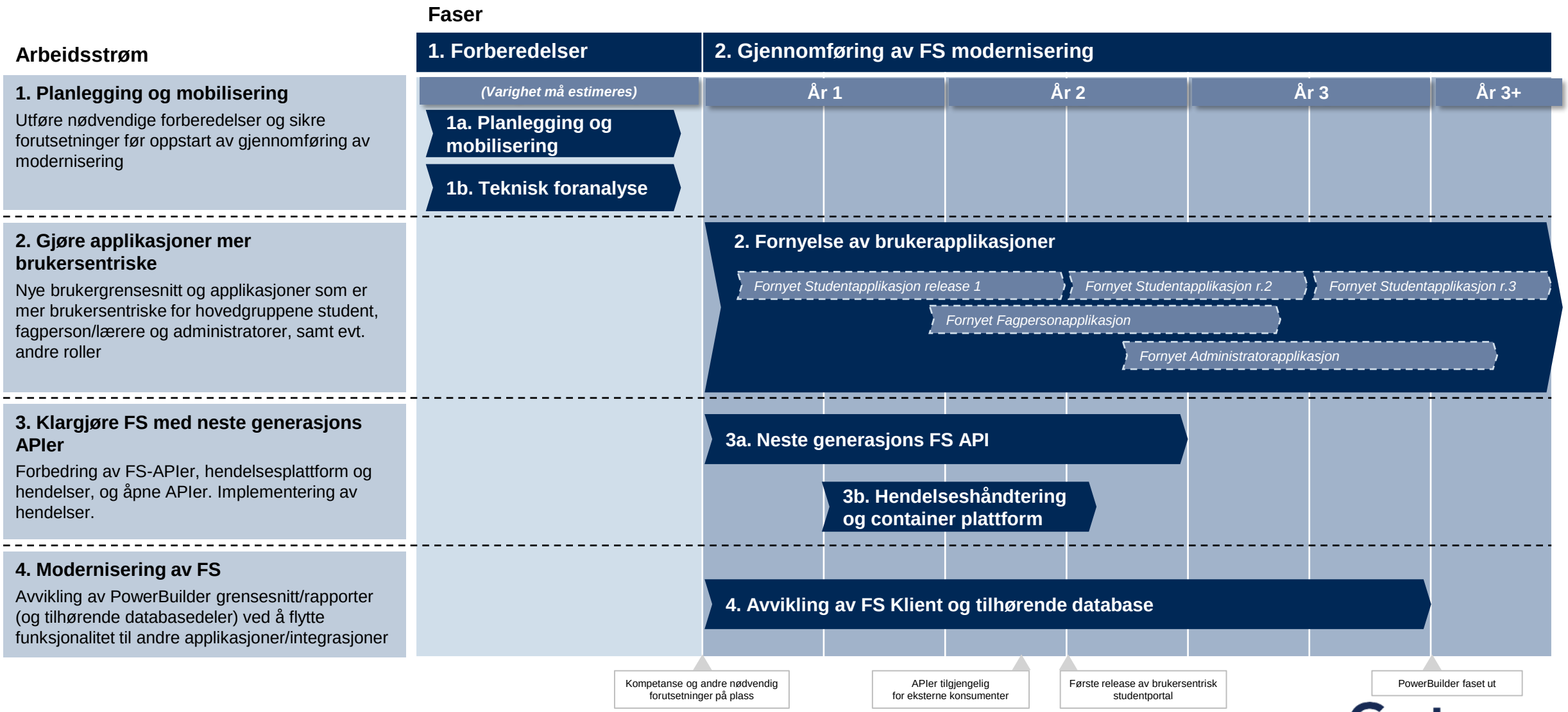
3. API utvikling (2 FTE)
- API-utviklere for å utvikle neste generasjons FS-API implementering, eksempelvis;
  - Legge til manglende API-er
  - Legg til use case APIer
  - Transformer eksisterende APIer til en ny tilgangshåndtering av DB
- API-teamet skal kunne bytte fokus til hendelseshåndtering og sikre implementering av hendelser

**Ressursestimatet er basert på en indikativ omfangsestimering av FS økosystemet, og med forutsetning at det er forbedringer av eksisterende løsninger og utgangspunkt i dagens behov og krav som finnes.*

3. Sluttrapport detaljer

3.1 Beskrivelser av veikartsaktiviteter

Treårig veikart for videre modernisering av FS mot målarkitekturen



1a. Planlegging og mobilisering

Formål	• Identifisere avhengigheter og andre krav/forutsetninger til gjennomføring (evt. hindringer) • Sikre nødvendig kompetanse og ressurser for gjennomføring

Oppgaver	Kompetanseplanlegging
	• Utarbeide plan for styrking av kompetanse/evne ved å rekruttere nye ressurser og/eller kompetansebygging av eksisterende ressurser innenfor
	  • Applikasjons- og løsningsarkitektur på tvers av teamene
	  • Teknologier for front-end, back-end og API-utvikling
	
	Samhandling og organisering for videre arbeid
	• Vurder samhandlingsform for FS-økosystemet for å sikre riktig og effektiv prioritering av krav, slik at et optimalt veikart kan oppnås og at duplisering av arbeid unngås
	• Analyser implikasjonene for forvaltningsorganisasjonen i denne utviklingsfasen
	  • Vurder et DevOps-team med produktorganisasjonsstruktur der det samme teamet administrerer både utvikling og drift
	
	Juridiske krav og avhengigheter til målarkitektur
	• Identifiser juridiske begrensninger og utforske mulige løsninger, GDPR-implikasjoner, dataeierskap og hvordan data kan nås og brukes i forskjellige brukergrensesnitt og implikasjonen for den generelle arkitekturen

Varighet	Varighet og innsats må planlegges og estimeres ved oppstart.
Notater	-

1b. Teknisk foranalyse

Formål	• Vurdere gjenbruk av eksisterende teknologier og løsninger for mest egnede oppstartspunkt for å utvikle persona-sentriske applikasjoner. Potensielle kandidater er: Vitnemålsportalen, en ny student web eller migrering av den gamle student weben • Identifisere avhengigheter og krav til målarkitektur

Oppgaver	• Utforme en persona-sentrisk applikasjonsarkitektur
	  • Utvikle passende målarkitektur for brukeropplevelser og back end-behandling (API-arkitektur vil bli administrert i API-arbeidstrømmen)
	• Velg passende teknologier og rammeverk for neste generasjons MASA-applikasjon (arkitektur av nett-apptjenester), for å også vurdere kompetansebehov
	• Utforske om hypotesen at student web er den mest egnet applikasjonen å starte å fornye
	  • Vurder hvor godt dagens brukersentriske vitnemålsportal kan støtte denne arkitekturen
	  • Vurder hva som er mest fornuftig, refaktor studentweb (for det meste riktig funksjonalitet men ikke riktig arkitektur) eller bygg fra grunnen av
	  • Vurdere en felles løsning for bruker-ID håndtering, som ikke krever at alle brukere skal være norsk statsborger, men også administrere at en student kan ha pågående eller tidligere studier ved flere institusjoner
	  • Identifiser manglende (om noen) APIer som trengs fra FS-API
	• Juster målarkitektur etter sektorsatsing og målarkitekturkrav - f.eks. bruk av skytjenester

Varighet	Varighet og innsats må planlegges og estimeres ved oppstart.
Notater	Undersøke hvor aktuell Vitnemålsportal arkitekturen er sammenlignet med ønsket tilstand, og hvis ikke, krever den en større gjennomgang av teknologier og rammeverk

2. Fornyelse av brukerapplikasjoner

Formål	• Utvikle brukersentriske applikasjoner

Oppgaver	Referanse- og løsningsarkitektur for webapplikasjoner
	• Gjenbruk av use case fra eksisterende applikasjoner, samt identifisere og implementere selvbetjenings use case
	  - Evaluer og velg hvilke use case som kan overføres «as-is» og hvilke som trenger merarbeid
	• Design og utvikling av nyttige gjenbrukbare komponenter
	• Utvikle en referansearkitektur for neste generasjons webapper (f.eks. ved bruk av React, Vue, Angular 2 osv.) basert på MASA (nettverksapplikasjon og servicearkitektur)
	
	APIer
	• Identifiser dupliserte API-er og konsolidere disse eller ta ut de gamle om nødvendig
	• Design og utvikle manglende resource og use-case baserte APIer
	• Implementere APIene ved å bruke «back-end for front end» designmønster for å understøtte brukeropplevelsene
	
	Database design
	• Opprett en persona-spesifikt database med data fra dagens relevante datatabellene i FS Database
	• Lag ID-er og transformer primærnøkler (slik at f.eks. en student ikke er avhengig av institusjon) og lag nødvendige referanser til andre FS-systemer
	
	Hendelseshåndtering
	• Formidle de aktuelle hendelsene fra studenttjenesten til FS-tjenesten for å sikre datakonsistens mellom studentweb og FS-applikasjonen
	  - Bruk «event-driven» arkitektur for å muliggjøre en så løs kobling som mulig mellom sammenhengende tjenester
	  - Kommuniser med tydelige forretningshendelser på høyt nivå (f.eks. Ny student, oppdatert kommunikasjonsprofil, innlagt på kurs osv.)
	  - Unngå synkron oppførsel (dvs. at tjeneste A legger ut en hendelse og trenger deretter en annen tjeneste for å legge ut en oppfølgingshendelse for å komme videre)

Varighet	• Estimert tid per applikasjon release (v1, v2..) er ca. 12-18 måneder i veikartet • Koordineres med aktivitetene i arbeidsstrømmen *3. Klargjøre FS med nestegenerasjons APIer*
Notater	▪ Det primære målet er å lage en ny studentapplikasjon som primært har de samme funksjonalitetene som den eksisterende studentweben, vitnesmålsportal og andre relevante applikasjoner for studenter, under analysen skal kravene verifiseres og kontinuerlige forbedringer av løsningene bør implementeres ▪ Det forventes ikke å starte fra bunnen av, men mye gjenbruk av eksisterende behov og krav

3a. Neste generasjons FS API

Formål	• Ferdigstille FS API arkitektur og et sett med APIer for bruksområder
Oppgaver	API Management plattform ▪ Vurdere de nødvendige funksjonene til en API-administrasjonsplattform ▪ Vurdere INTark valgt plattform (Gravitee) og sammenligne kapabiliteter, arkitektur krav og TCO mellom Gravitee, Mulesoft og evt. andre cloud baserte løsnigner ▪ Vurdere, anskaffe og implementere API Management plattform Neste generasjons FS API ▪ Lage designmønstre og retningslinjer for resource-, use-case og back-end for front-end APIer slik at det er klart hvordan APIer skal utvikles, hvordan gjenbrukbarhet kan økes og hvordan isolasjon mellom applikasjoner og konsumenten kan oppnås ▪ Lage designretningslinjer for hvordan du utfører parameterkontroll, forretningslogikk, forretningsregler etc. ▪ Klassifiser APIene basert på behov og gjenbrukspotensial ▪ Identifiser manglende API, både resource og use case basert på det ideelle veikartet av tjenester for å implementere i vitnemålsportal og studentweb alternativer ▪ Migrer FS-APIer til valgt API-plattform Åpne API ▪ Basert på krav og behov identifiser hvilke API-er som ville være fordelaktig ved å være åpne for publikum uten noen form for innlogging ▪ Evaluer behovet for en «read-only» infrastruktur for å håndtere skalerbarhet (ref. CQRS designmønster)
Varighet	Estimert gjennomføring over 24 måneder i veikartet
Notater	▪ Ferdigstill design av APIer for use case, dvs. hvordan man styrer RESTful API-er mot en use case eller handling i stedet for en CRUD-operasjon på en ressurs. Det er flere tilnærminger som å ha virtuelle ressurser eller legge til http-verb osv. ▪ Vurderer en ikke-java-basert tilnærming, for eksempel programmeringsspråket Go ▪ Evaluer om det er andre relevante designmønstre for API Management («throttling», trafikkstyring, inntektsgenerering, avanserte sikkerhetstiltak etc.) som kan være relevante og ikke er støttet av gravitee ▪ Der det er hensiktsmessig vurdere å bruke internasjonale standarder for datautveksling

3b. Hendelseshåndtering og container plattform

Formål	• Sikre at nødvendige forretningshendelser blir sendt gjennom denne plattformen • Etablere nødvendig tekniske kapabiliteter for hendelseshåndtering og container plattform
Oppgaver	Vurdere og implementere plattform for hendelseshåndtering • Vurder behov for tekniske kapabiliteter innenfor dagens hendelseshåndtering og fremtidige behov • Vurder om dagens plattform Apache kafka dekker behovene, og i så fall sørg for dimensjonering, implementering og nødvendige tiltak for å utnytte denne plattformen i den nye målarkitekturen Etablere plattform for Container Management • Implementere en «container management plattform» med funksjonalitet for «service mesh» Utarbeide designmøter for mikro-/minitjenester i Container miljøer • Utarbeide designmønsteret for «side car proxies» og utnytt «service-to-service» kommunikasjon • Vurder eksisterende mikro-/minitjenester mot dette designmønsteret og utfør evt. transformering til den
Varighet	Estimert gjennomføring over 12 måneder i veikartet
Notater	Forutsetter at nødvendig kompetanse er på plass og teknisk foranalyse er utført

4. Avvikling av FS Klient og tilhørende database

Formål	• Fjerning av PowerBuilder og tilhørende deler i FS-databasen, ved stegvis flytting, avvikling og fjerning av funksjoner i PowerBuilder FS klienten
Oppgaver	• Avvikle brukergrensesnitt, rapporter og dataelementer fra FS Klient og FS Database som ikke lenger er nødvendig • Erstatte VPD-funksjonaliteten for å separere institusjonsdata fra hverandre, og plassere data deretter etter ny tilnærming (eksempelvis «en institusjon per skjema») i riktig infrastruktur • Fjerning av alle VPD relaterte «triggers» for å sette kontekstkolonnene i FS Database
Varighet	• Gjennomføres over 36 måneder i veikartet
Notater	• Koordineres med de andre arbeidsstrømmene under gjennomføring • Dette innebærer ikke full omstrukturering av databasestrukturene i FS databasen, men fjerning av dataelementer (kolonner, tabeller, views og annet) ved flytting av funksjonalitet til nye tjenester. I den nye målarkitekturen har tjenestene sine egne databaser.

3.2 FS analyse i teknisk vurdering

(underlag på engelsk)

The overall rating of FS is quite on par with the expectation given the size, age and how it has been evolving

ISO Category	ISO Dimension	UI	API	Back end	ISO Dimension	UI	API	Back end
Performance efficiency	Time behaviour				Resource utilization			
	Capacity							
Compatability	Co-existence				Inter-operability	N/A		
Usability	Appropriateness	Out of scope			Learnability	Out of scope		
	Operability	Out of scope			End user error protection		N/A	N/A
	User interphase aestetics	Out of scope			Accessibility	Out of scope		
Reliability	Maturity				Availability			
	Fault tolerance				Recoverability			
Security	Confidentiality	Out of scope			Integrity		N/A	N/A
	Non-repudiation	Out of scope			Accountability	Out of scope		
	Authenticity	Out of scope						
Maintainability	Modularity				Reusability			
	Analysability				Modifiability			
	Testability							
Portability	Adaptability				Installability			
	Replaceability							

Rating: Ok | Improvements | Issues

Gartner

Summary
Performance efficiency

ISO Dimension	UI	API	Back end	Comments/Observations
Time behaviour				▪ Overall not built for scale ▪ Lack of caching ▪ Additional layers in e.g. stored procedures
Resource utilization				▪ Overall lack of micro service oriented services prevents efficient resource utilization of the API layer ▪ The multitenant structure in the database with triggers to ensure data access prevents efficient resource utilization
Capacity				▪ No concerns for the UIs ▪ The API design and lack of caching ▪ The database approach with stored procedures for API access and triggers to constrain information

Rating: Ok | Improvements | Issues

Gartner

Performance effiency / Time behaviour

Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

Findings

- There are multiple clients with different characteristics:
  - The PowerBuilder client is not likely to be time efficient due to the application architecture, complexity of logic and the immensely complex screens, however, the low user count per instance wouldn't be a big issue
  - The webb applications are not built for scale, the applications are not layered into cachable and reusable services and the applications local databases implies a clunky step wise local load before execution compromising data consistency
  - The APIs are typically resource oriented and in some cases not from a micro services capability stand point which drives a higher workload at the client

Suggested actions

- Increase caching in the API layer
- Gradual removal of PowerBuilder into more efficient web applications

Rating Improvements Layer APIs Back end

Performance effiency / Time behaviour

Degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements.

Findings

- There is no consistent measurement or logging of end to end processing times for integrations so no benchmarking can be made, however there are some concerns with the design:
  - The very heavy database PL/SQL layer to fetch data that also creates overly generic and not contextualized data e.g. person instead of student/fag person
  - The data volumes and amount of transactional update implies that caching should be used more broadly
  - The APIs are either:
    - Doing everything such as building up complex sql statements including jdbc without leveraging a connection pooling object
    - Or doing extremely layered factory/mapper/etc. java object structures that creates a tremendous amount of flexibility but also overhead
  - With the current payload the current inefficiency may not be of big concern but when exposing data it may

Suggested actions

- Removal of PL/SQL packages in the database and consolidate business logic in the API layer
- Creation of use-case based APIs with suitable actions

Gartner

Rating Improvements Layer UI APIs

Performance effiency / Resource utilization

Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

Findings

- The resource utilization is from a front end and API perspective is not a big concern. The application architecture is not so segmented/layered so each instance will drive some overhead during each instance
- The lack of micro service/component design and the multitude of design approaches in the client will drive increased use of infrastructure
- However the usage profile of the applications are quite controlled (limitations in number of students, institutions etc.) so it may not be a big issue right now

Suggested actions

- Creation of back end for front end services and more use case based APIs to optimize the back end workload
- Increase caching

Rating Improvements Layer Back end

Performance effiency / Resource utilization

Degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements.

Findings

- There is a tremendous amount of business logic and code executing in the database, a key challenge for databases due to its need to provide consistent, sequential logging in the Oracle LogWriter process. This typically constrains the database handler's ability to parallelize work efficiently. The key challenge with the current setup is:
  - Multi tenant in the same instance being separated by VPD driving unnecessary amount of traffic to the same processing threads / extent parallelization parameters
  - Multi tenant also drives the need for pre read trigger to enable filtering (which is unnecessary since there shouldn't be any cross institution use case)
  - The high complexity of procedures and functions to provide read from tables driving a generalization that may not be relevant

Suggested actions

- Remove PL/SQL API packages
- Consolidation of business logic to API layer
- Consider removing of VPD functionality
- Consider removing of additional business logic validation in triggers

Rating Ok Layer UI

Performance effiency / Capacity

Degree to which the maximum limits of a product or system parameter meet requirements.

Findings

- No major concern, the amount of logged on users are fairly low, except for peaks during e.g. opptak

Suggested actions

- N/A

Gartner

Performance effiency / Capacity

Degree to which the maximum limits of a product or system parameter meet requirements.

Findings

- The APIs does show some technical design issues from a capacity point of view (decreases maximum capacity)
  - The lack of composite – use case driven services, instead there is the need for multiple resource based services which causes a capacity cap since it moves processing and invocation of additional APIs
  - The absence of caching, which could be leveraged due to the overall low update frequency of data

Suggested actions

- Implement back end for front end services and use case oriented service to optimize performance
- Implement more caching

Rating Improvements Layer Back end

Performance effiency / Capacity

Degree to which the maximum limits of a product or system parameter meet requirements.

Findings

- The database is most likely to be a bottleneck through the amount of processing to manage:
  - Authority to access data using the VPD and additional triggers
  - Processing and prepackaging data to the API layer rather than just shipping raw dataset and let the API layer manage it
  - Data validation, e.g. checksum verification, this should be done in the entry point and give instant feedback to the user if incorrect rather than resulting in a failed update transaction

Suggested actions

- Remove PL/SQL API packages
- Consolidation of business logic to API layer
- Consider removing of VPD functionality
- Consider removing of additional business logic validation in triggers

Summary
Compatability

ISO Dimension	UI	API	Back end	Comments/Observations
Co-existence				▪ No/few technical constraints of adding other applications or capabilities to each layer in the stack ▪ Fairly well controllable volumes and workload
Inter-operability	**N/A**			▪ Overall lack of micro service oriented services prevents efficient resource utilization of the API layer ▪ Scattered or lacking of documentation ▪ The information architecture is well developed ▪ Some database (API) services are creating a complexity by merging objects

Rating: Ok | Improvements | Issues

Gartner

Rating Ok

Layer UI APIs

Compatability / Coexistence

Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

Findings

- Should be fine. The servers have few constraints and appear to only leverage standard Linux + Jboss environments and shouldn't impose any great constraints from a coexistence perspective. The question is rather if there is any free infrastructure to leverage

Suggested actions

- N/A

Gartner

Rating Ok Layer Back end

Compatability / Coexistence

Degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

Findings

- The database servers has no technical constraint of adding additional capabilities or applications should carefully analyze the consequence of adding other applications or services to them due to the expected utilization rate and that FS seems be less performance efficient

Suggested actions

- N/A

Compatability / Interoperability

Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Findings

- Not really applicable for front end

Suggested actions

- N/A

Gartner

Compatability / Interoperability

Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Findings

- Interoperability can be divided into three categories:
  - Discoverability or access of the service, this is limited due to needed logons and tokens. There is not a good and simple process of managing these
  - Documentation of API/service descriptions (methods, parameters etc.) here the FS-API is well documented using swagger but the FS-REST are not as well documented
  - Putting services in a context. This is borderline impossible due to several reasons:
    - They are resource based with no “actions” associated with them
    - They require a complete understanding of the taxonomy and ontologies in place to ensure that they make sense

Suggested actions

- Continue and strengthen the use of Swagger for API documentation / code generation
- Enhance the data modelling skills into information modelling and API management and integration architecture

Compatability / Interoperability

Degree to which two or more systems, products or components can exchange information and use the information that has been exchanged.

Findings

- Interoperability can be divided into three categories:
  - Discoverability is by nature more difficult in the database due to lack of good standards to promote these
  - Documentation of the packages allowing access to the database is lacking
  - There is a good information architecture (but complex) to understand. Some services are however creating a potentially complexity by aggregating objects such as student and fag-person into a combined person object, that may have a limited use

Suggested actions

- Continue the work on good information architecture practices

Summary
Usability

ISO Dimension	UI	API	Back end	Comments/Observations
Appropriateness	**Out of scope**			
Learnability	**Out of scope**			
Operability	**Out of scope**			
User error protection		**N/A**	**N/A**	▪ Limited and sometimes lacking of validation or range checks of input parameters, these are typically caught by the database
User interphase aestetics	**Out of scope**			
Accessibility	**Out of scope**			

Rating: Ok | Improvements | Issues

Gartner

Usability / User error protection

Degree to which a system protects users against making errors.

Findings

- There are multiple applications each with different approaches to manage user entries and validation. The entry fields validations are kept to a minimum

Suggested actions

- Develop design guide lines on what types of user error protection that makes sense
- Develop mechanisms for automating implementation of field validation using components, configurable templates, integration of information architecture tool to generate these templates etc.

Summary
Reliability

ISO Dimension	UI	API	Back end	Comments/Observations
Maturity				▪ The PowerBuilder is still supported but has passed the «dusk of obsolence» ▪ Oracle announced the Java EE being handed over to open source community and this indicates that this platform is not seen as future proof for the long term ▪ The API layer is quite inconsistent in its architecture and design
Availability				▪ Front end applications runs on load balanced servers ▪ APIs are running on load balanced servers and in some cases on API platform (Mule) ▪ The database utilize cluster or data replication capabilities
Fault tolerance				▪ Each application relies intimately on underlying APIs and compoents so each layers fault tolerance is impacted by lack of clustering (e.g. Of the database) or caching
Recover-ability				▪ Strict transaction management and stateless clients prevent data from being partially persisted ▪ Lack of inbound queues or file management implies that the API layer doesn't need any particular recover routines ▪ The back end has dependency to the RabbitMQ that can stop but it shouldn't affect the database server

Rating: Ok | Improvements | Issues

Gartner

Reliability / Maturity

Degree to which a system, product or component meets needs for reliability under normal operation.

Findings

- The applications are built using four key groups of technologies
  - PowerBuilder for “fat client usage”
  - JSF for web apps
  - React for web apps
  - Mobile apps
- PowerBuilder has passed the dusk of obsolescence and there is also a risk with regards to resources should be replaced
- JSF relies on Java EE and application servers and that technology has been communicated by oracle that it is about to be abandoned

Suggested actions

- Initiate gradual decommissioning of the PowerBuilder functionality
- Evaluate options to Java EE for business logic in the middle tier
- Decommission the JSF applications to new the proposed applications

Reliability / Maturity

Degree to which a system, product or component meets needs for reliability under normal operation.

Findings

- The approach of building REST interfaces towards well described resources is well in line with best practices, the question is to what extent the multi layered java on top on several layered PL/SQL packages impacts:
  - Good performance with the multi layered multi object structure to fetch data through factories, mappers etc.
  - Easy maintenance since change of datatype now must be changed in the table, PL/SQL package, several java components and finally the API signature
- The API approach is not fully inline with expected best practices in terms of caching, security, traffic management, robustness and the API layer(s) is not fully developed/consistent

Suggested actions

- Implement the FS-API next generation with back end for front end and use case oriented patterns
- Decommission PL/SQL packages for APIs
- Implement API management platform

Rating Ok Layer Back end

Reliability / Maturity

Degree to which a system, product or component meets needs for reliability under normal operation.

Findings

- There are no unproven, immature or legacy technologies used in the back end
- There are however design paradigms such as:
  - the great usage of triggers and PL/SQL that can be questioned from a maturity perspective since a majority of the solutions has abandoned these designs due to the constraints it drives
  - Extensive use of referential integrity creates strict requirements on presence of other data, potentially in other domains creating a tighter coupling between data entities we would prefer a much more loosely coupling

Suggested actions

- N/A

Reliability / Availability

Degree to which a system, product or component is operational and accessible when required for use.

Findings

- Front end applications are load balanced and runs on individual computing environments

Suggested actions

- N/A

Gartner

Rating Ok Layer APIs

Reliability / Availability

Degree to which a system, product or component is operational and accessible when required for use.

Findings

- The APIs are both load balanced and runs on different types of platforms, everything from Mule to Spark

Suggested actions

- N/A

Rating Ok Layer Back end

Reliability / Availability

Degree to which a system, product or component is operational and accessible when required for use.

Findings

- The database runs in a failover mode using DataGuard to ensure data synchronization

Suggested actions

- N/A

Gartner

Reliability / Fault tolerance

Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

Findings

- Overall fault tolerance is impacted by the lack of caching and alternative approaches to unavailability of underlying services

Suggested actions

- Implement service mesh like functionality and implement back end services in a more robust way, e.g. improve the stateful-ness, self-recoverable services etc.

Rating Improvements Layer APIs Back end

Reliability / Fault tolerance

Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.

Findings

- There is a load balancing for the API layer
- The strict dependency to the database creates a major dependency, especially since the FS database has such a vast dataset
- There are also dependencies to the messaging component that creates issues if it is not being emptied, it can literally stop

Suggested actions

- Implement API management platform
- Decrease the tight coupling between the APIs and back end by e.g. removing the PL/SQL PAI packages

Rating Ok Layer UI

Reliability / Recoverability

Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

Findings

- There is no state kept in the client, everything is persisted so there are no real recovery issues
- Strict transaction management cannot leave data in partial state and as such not in the need for automated recovery

Suggested actions

- N/A

Rating Ok Layer APIs Back end

Reliability / Recoverability

Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

Findings

- Recovery in the API and back end is easier due to lack of inbound queues or usage of files for e.g. FTP purposes
- The only component that imposes a risk is the RabbitMQ component that can stop if the queue gets full. The writing to the queue goes through a database table so no messages should be lost
- The database has routines to recover pending non written transactions to the database. The strict transaction management and referential integrity prevents data from being non-consistent

Suggested actions

- N/A

Summary
Security

ISO Dimension	UI	API	Back end	Comments/Observations
Confidentiality	**Out of scope**			
Integrity				▪ The database ensure using VPD and triggers that no other data than the current institution is being showed ▪ Token authorization ensures the access rights of invoking the API
Non-repudiation	**Out of scope**			
Accountability	**Out of scope**			
Authenticity	**Out of scope**			

Rating: Ok | Improvements | Issues

Gartner

Rating Ok

Layer UI

Back end

Security / Integrity

Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Findings

- There is a bespoke authorization mechanism within the applications to ensure that you only access the data you should and the database VPD mechanism ensures that you're always in the right dataset
- The approach to build up SQL statements prevents e.g. SQL Injection

Suggested actions

- N/A

Gartner

Rating Ok Layer APIs

Security / Integrity

Degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data.

Findings

- There is a token based approach to authorize the invocation of the API but no real business logic within the API itself

Suggested actions

- N/A

Summary
Maintainability

ISO Dimension	UI	API	Back end	Comments/Observations
Modularity				▪ The UI's are not fully domain or service centric in its design ▪ The APIs are not bundled into domains and don't utilize the back end for front end to improve cohesion ▪ The database doesn't follow any real domain model and uses queries across any table and heavy use of referential integrity
Reusability				▪ The UI layer is not designed with reuse as a design principle (e.g. In the React space) ▪ The APIs are in some cases highly reusable (FS-API), but these are without the BFF pattern more complex to put in a useful context. Many other APIs are highly specific to the usecase without using underlying reusable services ▪ Database wise the reusability is higher, but on the other hand moving over into a more complex and abstract type of service
Analysability				▪ Lack of documentation and large numbers of technologies and design principles ▪ The multiple layers doing similar things and implementation of business logic ▪ Lack of tools that can analyze and detect dependencies and usage
Modifiability				▪ Lack of reusable components ▪ Multiple overlapping APIs ▪ The database with the strict referential integrity and lack of domain segmentation
Testability				▪ There is an overall lack of implementation of automated testing, only partial coverage of unit tests ▪ Performance testing is not utilized ▪ The complexity of some services are so high that auomated testing will be difficult

Rating: Ok | Improvements | Issues

Gartner

Rating Improvements Layer UI

Maintainability / Modularity

Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Findings

- The modularity is different in different parts of the landscape
- PowerBuilder apps are intimately connected to the underlying database without a service layer, the functions
- There is overall not a domain or service centric design
- The react applications are well structured into different components but not decomposed into reusable components
- Each application is also developed from a very specific persona/use case not delivering components or services as stand alone modules that could be reused

Suggested actions

- Implement FS-API next generation APIs with improved architecture, additional APIs, additional back end for front end services
- Improve usage of components in the front-ends with the to-be architecture

Maintainability / Modularity

Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Findings

- The modularity in the API layer is of different level of clarity and cohesion:
  - FS-REST is a mixture of what appears to be more generic and reusable services and at the same time several services are strictly developed for the consuming application
  - FS-API which is highly revolving around high level business entities. Some that are so generic and distant from relevant use cases
  - FS-WS (SOAP) is only maintained when absolutely necessary, these are planned to be decommissioned May 2020
  - The back-end for front end pattern is not used that could improve modularity

Suggested actions

- Implement FS-API next generation APIs with improved architecture, additional APIs, additional back end for front end services
- Ensure that FS-API covers the old APIs and change the front ends accordingly

Gartner

Maintainability / Modularity

Degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Findings

- The backend has a poor modularity, due to:
  - The vast size of FS-DB
  - The strict usage of referential integrity referencing across domains
- There are several other “local databases” in the landscape e.g. EpN etc. these are stand alone and does allow for the applications to manage the data independently but at the same time, data is partially copied to these are intended to be written back, without locking. This creates a complicated and potentially erroneous situation. Modularity implies ownership and that should not be shared.

Suggested actions

- Disentangle the database into more micro-/mini-services

Maintainability / Reusability

Degree to which an asset can be used in more than one system, or in building other assets.

Findings

- Reusability is not the main driver for UI development
- No real reusable assets are being developed in the react space, there are a greater degree of reuse in the JSF applications

Suggested actions

- Design a more component based approach for the user interface and improve the amount of reuse

Maintainability / Reusability

Degree to which an asset can be used in more than one system, or in building other assets.

Findings

- The API's can be divided in two categories:
  - Resource based (REST, FS-API) that are highly reusable with the caveat that it requires a significant understanding of the taxonomy and ontology in order to consume them in a productive way
  - Single consumer focused, i.e. developed with the focus of streamlining the work for the consuming application i.e. no reuse in mind, this is predominantly the FS-WS approach
- The lack of aggregated services e.g. using the back end for frontend pattern drives more tightly coupled services bespoke for the consumer

Suggested actions

- Implement FS-API next generation APIs with improved architecture, additional APIs, additional back end for front end services

Maintainability / Reusability

Degree to which an asset can be used in more than one system, or in building other assets.

Findings

- The back-end is well encapsulated for API consumption using the stored procedures, there are however some concerns from a reusability perspective:
  - There are business logic in both the stored procedures (ensuring referential integrity), triggers (validations) and constraints (referential integrity primarily)
  - There are usage of object oriented principles that creates a level of abstractions that can create constraints or side effects in the clients where the use case most likely is not in line with these abstractions (e.g. get person, this will in most clients either be getStudent or getFagperson

Suggested actions

- Move business logic from the database to the APIs

Maintainability / Analysability

Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

Findings

- The Analysability is hampered due to multitude of reasons:
  - Overall PowerBuilder complexity due to the PowerBuilder application structure with all the components and events and the fact that the tool is old and lacks of market traction
  - The webapps doesn’t follow a consistent solution architectural approach and some use more clearly defined APIs from FSAPI and some do plain SQL as being executed through jdbc
  - The overall lack of documentation
  - The large number of technologies, design principles and solution architectures

Suggested actions

- Improve standardization of how UIs are developed
- Improve overall documentation

Maintainability / Analysability

Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

Findings

- The overall analyzability of the API layer is fairly low due to:
  - Lack of documentation (not for consumption purposes but rather the internal design and description) (this is not the case for the MuleSoft integrations)
  - Large number technologies
  - Lack of consistency in solution architecture (the Java EE structure in the API layer, how SQL are being generated and how it is invoked)

Suggested actions

- Improve standardization of how APIs are developed
- Improve overall documentation

Maintainability / Analysability

Degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.

Findings

- Analyzability is hampered due to multitude of reasons:
  - The size of the database structure (amount of tables)
  - The amount and complexity of PL/SQL objects both for API purposes, other business logics (primarily “routines”), triggers to ensure non-mutating execution and performing business logic
  - The lack of good documentation integrating all aspects of business logic operations

Suggested actions

- Improve overall documentation

Gartner

Maintainability / Modifiability

Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Findings

- The UI is due to its separation into “use case groups of apps” i.e. an app for a specific purpose (or set of use cases) makes it easy to do specific changes to that application without risking dependencies (since the lack of reusability)
- However, the lack of reusable components makes cross application or more fundamental requirements significantly more difficult to implement e.g. change of person name length/structure etc.
- There are some dependencies in the JSF applications to Jquery and other legacy libraries that constrain maintenance

Suggested actions

- Improve standardization of how UIs are developed
- Improve overall documentation

Maintainability / Modifiability

Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Findings

- The API layer is from a technical point of view fairly easy to modify, either the API:
  - Is well structured and layered and has a consistent architecture
  - Is a “monolith” like the FS-WS APIs where everything is in one box and can easily be managed with few dependencies
- The functional overlap between APIs of the different types prevent easy modifications of cross application requirements
- The lack of usage insight or documentation (i.e. API management) makes it more difficult to modify the FS-APIs and life cycle management needs to be implemented

Suggested actions

- Improve standardization of how APIs are developed
- Improve overall documentation

Maintainability / Modifiability

Degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Findings

- Modifiability is hampered due to multitude of reasons:
  - The size of the database structure (amount of tables)
  - The table dependencies in terms of referential integrity
  - The amount and complexity of PL/SQL objects with business logic and event driven code (triggers) that makes it difficult to get an overview how the system actually works

Suggested actions

- Start dis-entangeling the database into micro-/mini-services
- Improve overall documentation

Maintainability / Testability

Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Findings

- The diversity of client technologies and lack of testing strategies makes it difficult to implement automated testing
- From a technical perspective automated testing in PowerBuilder is more complex than for java/react applications
- Testability is hampered due to the lack of modularity and cohesiveness of the functions. It does become somewhat more challenging to create automated testing for the user interfaces

Suggested actions

- Improve/implement automated testing capabilities

Maintainability / Testability

Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Findings

- Testing of APIs doesn’t utilize fully automated testing tools and neither is monitoring from a performance perspective there is partial unit test coverage and API loggin
- The APIs should be able to be tested using automation technologies due to their nature of having well encapsulated and cohesive functionality, especially for FS-API, the FS-WS APIs it could be more challenging
- The MuleSoft APIs are, in comparison, being monitored and logged in a more structured way using logs and kibana

Suggested actions

- Improve/implement automated testing capabilities

Maintainability / Testability

Degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Findings

- The back-end is cohesive in the sense that packages contain functions for the same type of object or routine, as such automated testing should be leveraged
- However, stored procedures and database routines are normally not the most automated area of testing

Suggested actions

- Improve/implement automated testing capabilities

Summary
Portability

ISO Dimension	UI	API	Back end	Comments/Observations
Adaptability				▪ No concerns
Installability				▪ No concerns for all areas except the PowerBuilder client that has restrictions on the target environment, currently this is resolved using terminal servers which is a cumbersome and cost driving solution
Replaceability				▪ PowerBuilder and Oracle will be highly complex to replace in the sort term ▪ The API and web clients are way easier to replace due to the technologies available

Rating: Ok | Improvements | Issues

Gartner

Rating Ok Layer UI

Portability / Adaptability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The web applications are running on platforms (Jboss EAP etc.) that are highly portable, they seem not to leverage the Jboss platform too much and it could easily take advantage of other services or components
- The PowerBuilder client however only runs on Windows and to resolve dependencies to the existing client computers UNIT leverage a terminal server solution. This works well from a technical standpoint but introduces new infrastructure, licenses and cost

Suggested actions

- N/A

Rating Ok Layer APIs

Portability / Adaptability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The API layer is either running in application servers without dependencies to the application server specific capabilities or in docker containers

Suggested actions

- N/A

Rating Ok Layer Back end

Portability / Adaptability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The backend is so tightly connected and dependent on Oracle which should be of less concern given the amount of supported platforms and hardware types

Suggested actions

- N/A

Portability / Installability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The PowerBuilder client however only runs on Windows and to resolve dependencies to the existing client computers UNIT leverage a terminal server solution. This works well from a technical standpoint but introduces new infrastructure, licenses and cost

Suggested actions

- Phase out PowerBuilder by moving functionalities to the other applications and implement an administrator application

Portability / Installability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The APIs can run anywhere a Jboss AppServer or any compatible appserver can be installed, so this is no concern

Suggested actions

- N/A

Rating Ok Layer Back end

Portability / Installability

Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments.

Findings

- The Oracle DB has runtime for several platforms and no OS specifics are in place

Suggested actions

- N/A

Portability / Replaceability

Degree to which a product can replace another specified software product for the same purpose in the same environment.

Findings

- The PowerBuilder client is built from a resource perspective and not as many other 4GL tools driving a certain design in the database, as such the replaceability of front end is not technically constrained

Suggested actions

- Phase out PowerBuilder by moving functionalities to the other applications and implement an administrator application

Portability / Replaceability

Degree to which a product can replace another specified software product for the same purpose in the same environment.

Findings

- Since the majority of the APIs are REST (in some flavor) they are easily replaced with another compliant interface
- The lack of platform dependencies makes it fairly easy to replace the different components

Suggested actions

- N/A

Portability / Replaceability

Degree to which a product can replace another specified software product for the same purpose in the same environment.

Findings

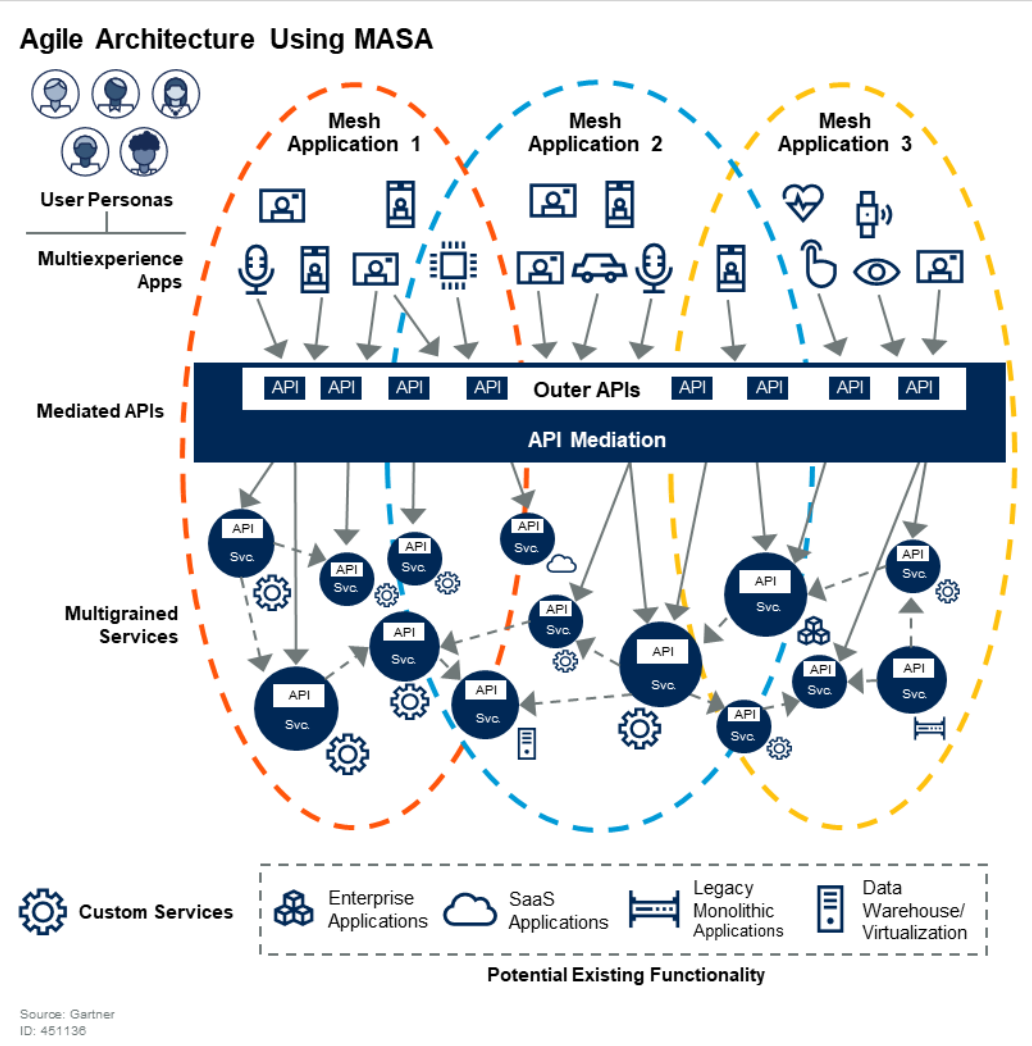
- To replace Oracle with another database is in reality impossible with the current amount of PL/SQL code and the amount of business logic that is shifted down in the stack instead of being managed higher up

Suggested actions

- Remove the PL/SQL API packages
- Disentangle the database to micro-/mini-services
- Move business logic from the database to the API layer

3.3 Relevant WS materiale for veikartsaktiviteter

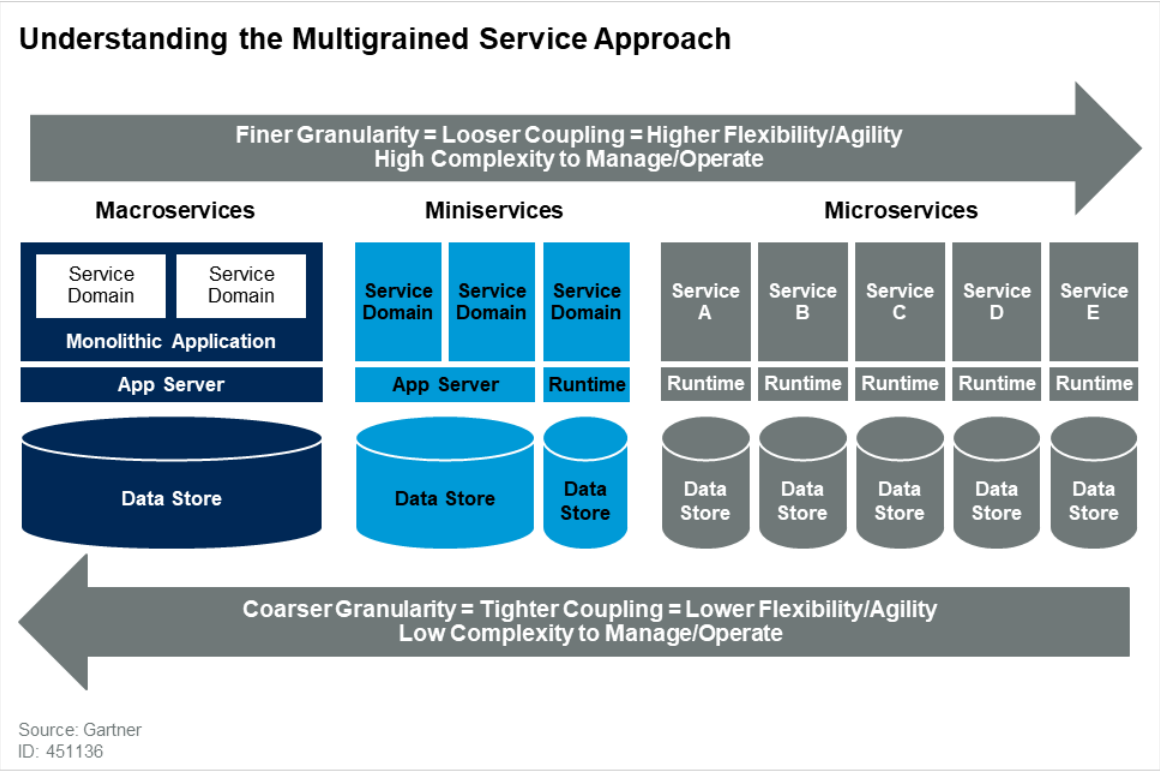
Evolve EpN, Student Web etc. towards a MASA architecture



- Start evolving the existing web applications and apps towards a mesh up architecture where the apps gravitate towards user personas instead of processes
  - E.g. student app, FagPersonWeb, Administrator WEB etc. are relevant starting points for applications for distinct personas
- Define the proper services to be leveraged in the different apps, e.g. back-end for front-end APIs towards more reusable outer APIs
- Coordinate the future roadmap and plan with sector initiatives to replace relevant functionalities e.g. söknad/opptak

To read about it, Gartner Research: https://www.gartner.com/document/3980382

Disentangle the DB monolith

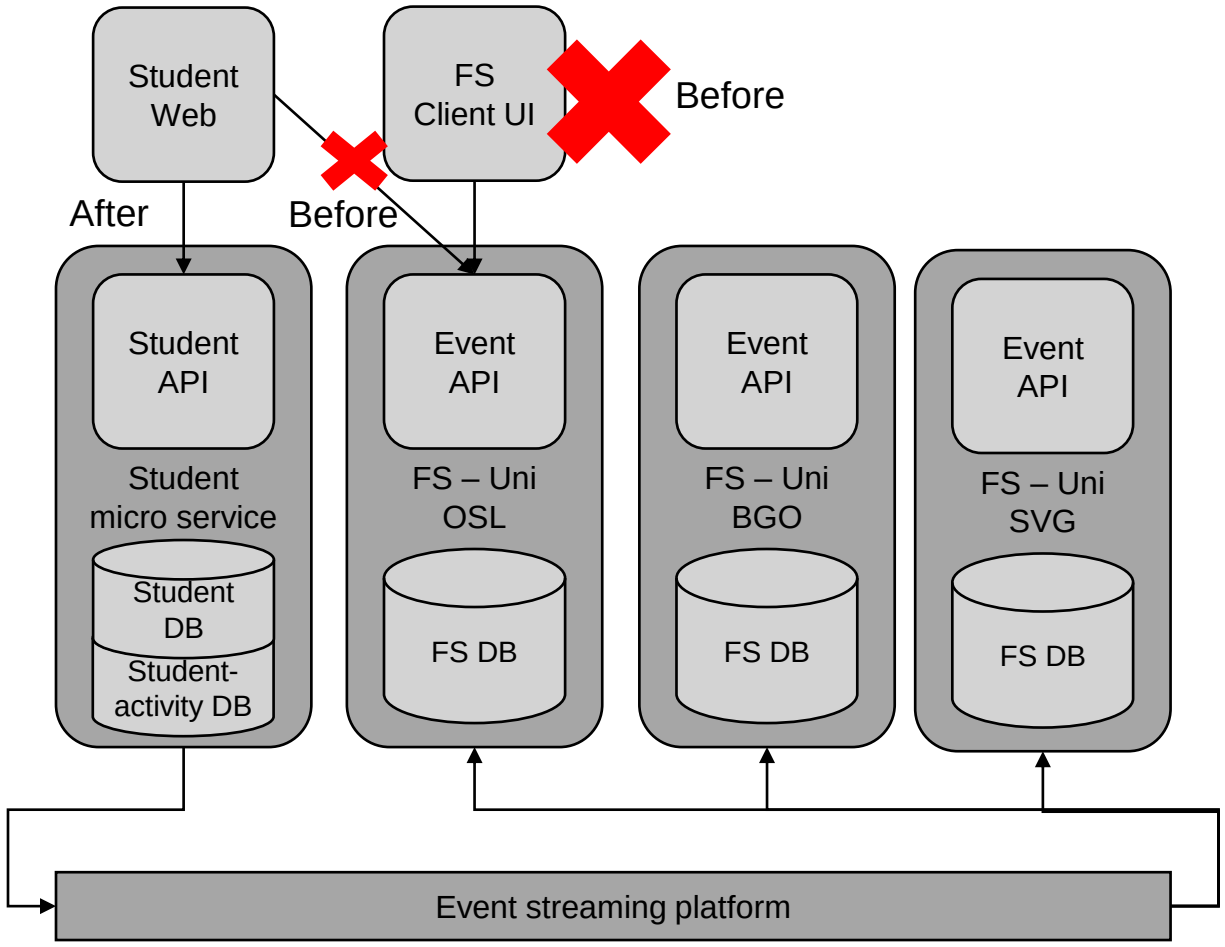


- A consequence of a large monolithic database is that:
  - It drives dependency chains that can be cumbersome to maintain consistency over time and worst case starts to impact the solutions functionality
  - It allows us to “cheat” and consume data from any other data source in the database
- A key design feature of micro services is that each service maintain its own data, in the way it desires and using the most suitable persistency mechanism available
- Yes, this approach will duplicate data, it does rely on eventual consistency and for many of the use cases this is good enough. On the other hand the increased flexibility typically increases for the individual teams if they can focus on their own domain

To read about it, Gartner Research: https://www.gartner.com/document/3980382

Gartner

Disentangle the DB monolith, by breaking out relevant domains of information



- Identify and break out the relevant domains of the domain entities that makes sense to carve out, some candidates are:
  - Student (should be institution agnostic)
  - Results (should be student centric)
- Establish a micro service for these domains with its own APIs and data persistence. Student could easily benefit from a document database approach and have a more loose data model
- Implement an event streaming platform to propagate relevant events on the information objects so that the Event API in FS DB can implement the relevant information items
- Increase the self service ratio of functions in the student web
- Increase the automation of tasks using data and services
- Decommission the FS Powerbuilder UIs and reports that are now maintained elsewhere or doesn't have a FS specific use case
- Redirect student web API calls to the new micro service

Gartner

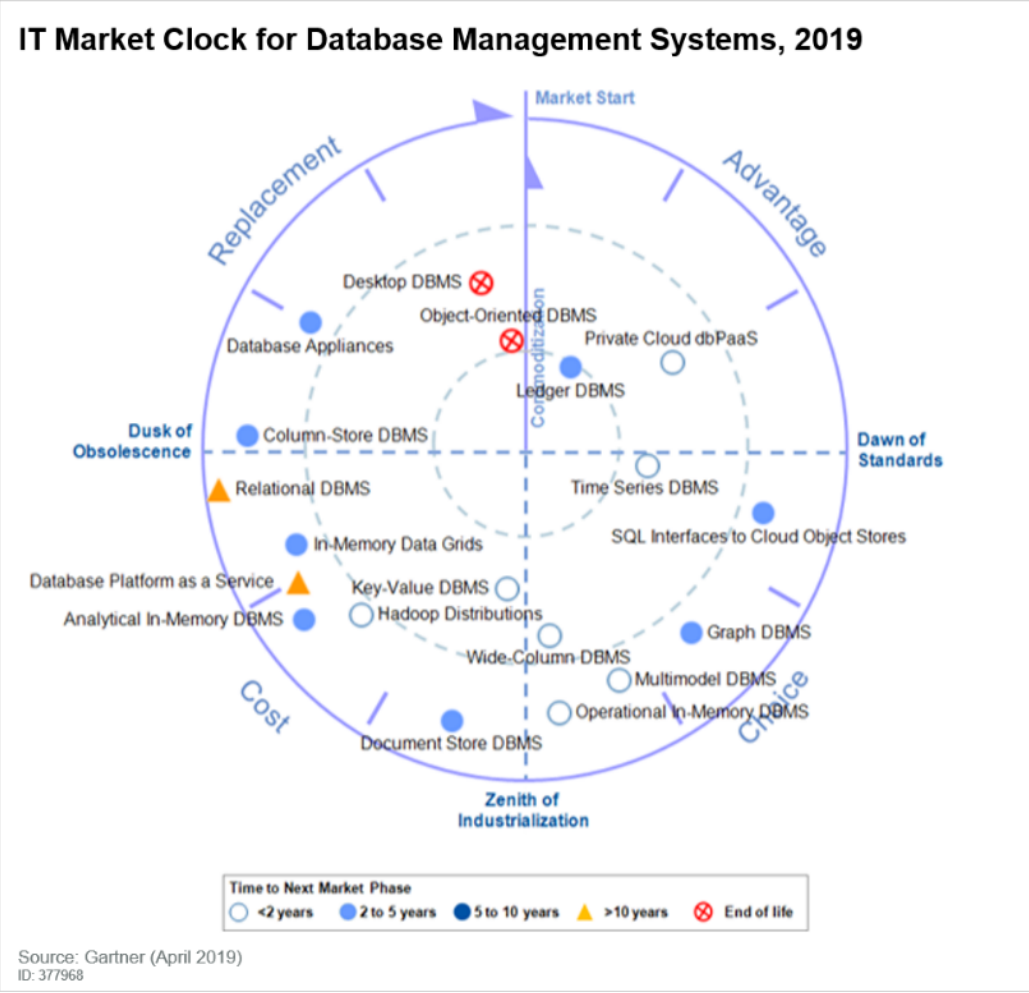
Disentangle the DB monolith, by breaking out relevant domains of information and streamlining what data is needed in FS

```
STUDENT
P  Fodselsdato                 (Person)              N(6)     NN
P  Personnr                    (Person)              N(5)     NN
   Studentnr_Tildelt                                 N(6)
   Adrlin1_Semadr                                    VC(60)
   Adrlin2_Semadr                                    VC(60)
   Postnr_Semadr                                     N(4)
   Adrlin3_Semadr                                    VC(30)
   Adresseland_Semadr                                VC(30)
   X_Telefonlandnr_Semtelefon                        VC(3)
   X_Telefonretnnr_Semtelefon                        VC(5)
   X_Telefonnr_Semtelefon                            VC(8)
   Adrlin1_Arbeid                                    VC(60)
   Adrlin2_Arbeid                                    VC(60)
   Postnr_Arbeid                                     N(4)
   Adrlin3_Arbeid                                    VC(30)
   Adresseland_Arbeid                                VC(30)
   X_Telefonlandnr_Arbeid                            VC(3)
   X_Telefonretnnr_Arbeid                            VC(5)
   X_Telefonnr_Arbeid                                VC(8)
   SPRAKKODE_VITNEMAL          (Sprak)               VC(10)
   Fichekode                                         VC(10)
   Merknadtekst                                      VC(1000)
   Bibsyslanetakerid                                 VC(20)
   Status_Hjemmeleser          (   J/N   )           VC(1)
   Status_Relegert             (   J/N   )           VC(1)
   Arstall_Relegert_Til        (Ar_Arstermin)        N(4)
   TERMINKODE_RELEGERT_TIL     (Ar_Arstermin)        VC(4)
   TROSGRUPPEKODE_TILKNYTTET   (Trosgruppe)          VC(10)
   SAKSBEHINIT_OPPRETTET       (Saksbehandler)       VC(10)
   Dato_Opprettet                                    D
   SAKSBEHINIT_ENDRING         (Saksbehandler)       VC(10)
   Dato_Endring                                      D
   Dato_Endret_Semadr                                D
   KUNDEGRUPPEKODE             (Kundegruppe)         VC(10)
   Utskriftskode                                     VC(20)
P  Institusjonsnr_Eier         (Person,
                                Trosgruppe,
                                Kundegruppe,
                                Bibsysbeststed,
                                Ar_Arstermin,
                                Saksbehandler)       N(8)     NN
   BIBSYSBESTSTEDKODE          (Bibsysbeststed)      VC(10)
```

- Go in detail and see what are the functional needs for information within the remaining functionalities of the core-FS
- E.g. The student table appears to mostly contain address, contact, phone information that is probably of more interest to the student service
- Consider implementation of a Student.Communicate() API where any kind of communication from FS could be sent to that service for further processing (based on e.g. student preferences), it doesn't have to be done from FS

Gartner

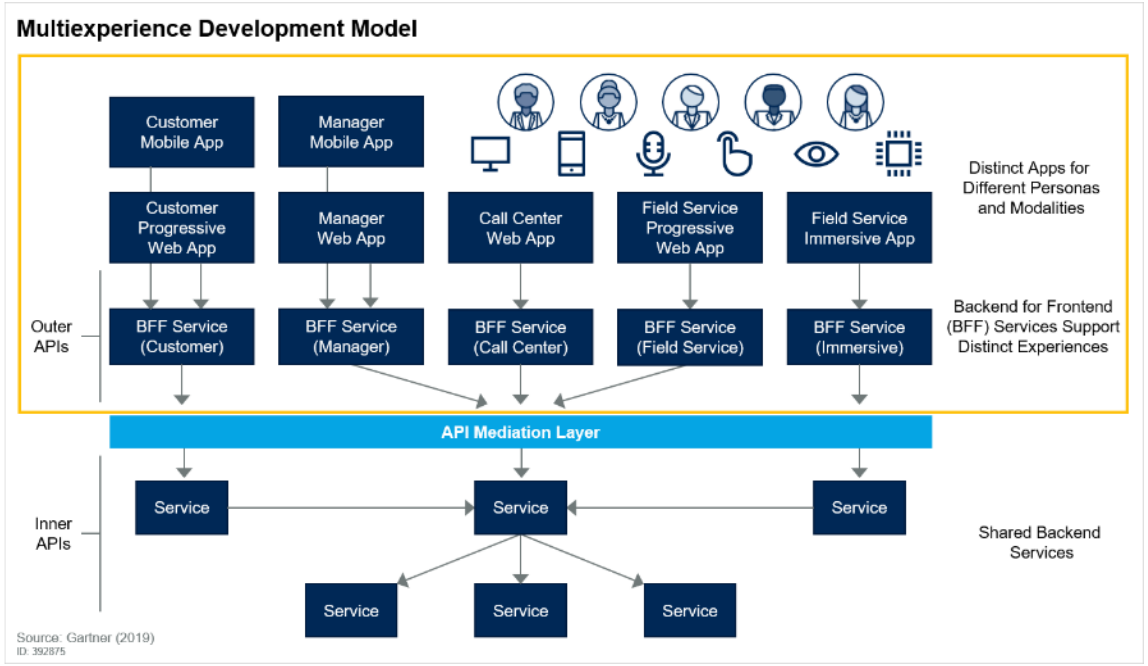
Assess the most suitable persistence technology per service

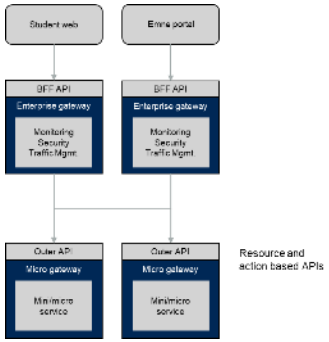


- Relational databases are still having a dominant market position and they are classified as more than 10 years before they are considered as becoming obsolescence
- However, the “new” no-sql type of databases are getting increased traction, they perform and scale way better than before and they map very well to some of the design concepts that are more popular now e.g. micro services
- One example would be student centric data (i.e. all contact information, study results, current activities). What would the benefit be to have these in a few tables blended with other students data? Instead of having a document with all relevant data for that student?
- Former problems like de-normalization, storage constraints etc. are less concern of today and many commercial systems anyway make a full copy of the customer data on e.g. a customer order since the next customer order may have another address

Gartner

Insert a use case type of services and/or a Back end For Front end type of services

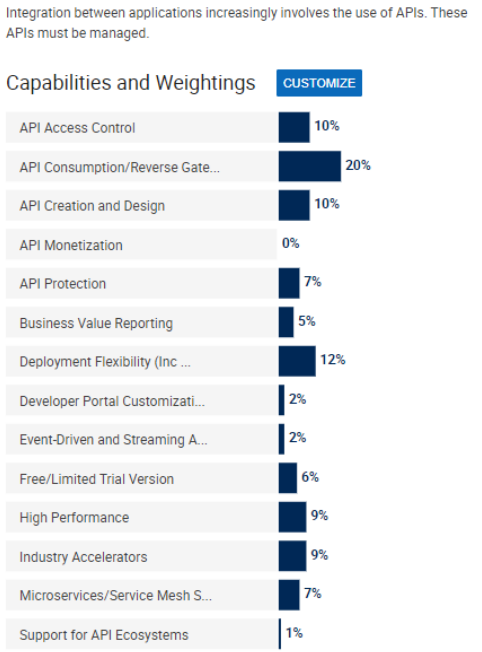


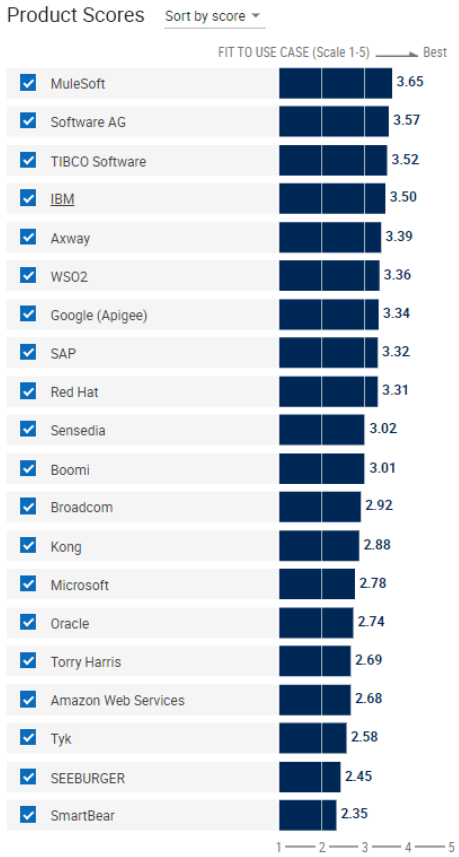


- Today's resource APIs are fine and doesn't need any interface type of change, they do reflect the business objects in the data model
- There are two extensions being proposed:
  - Identification and implementation of the use case types of services being implemented in the different applications. E.g. Student applies for a emne, someone needs to validate if the person is an active student.
    These services typically involve a multiple set of resources and some kind of business logic (when do we consider the student to be an active student)
  - Implementation of a back end for front-end services i.e. tailoring of use case services that are potentially a little bit to broad or send too much data to the consumer. This applies in particular when there are multiple consumers of the same use case and they are significantly different in their capabilities, e.g. a smart phone vs desk top client
- Back end for front-end enables the front end from not having to deal with overly many and complex API calls
- BFF APIs provide caching and offloads the client from more complex processing such as nested service calls
- BFF's must balance the level of specialization versus reuse of services to not overly drive complexity and maintenance work

Gartner

Assess gravitee vs other API management platforms to better understand which platform that makes most sense

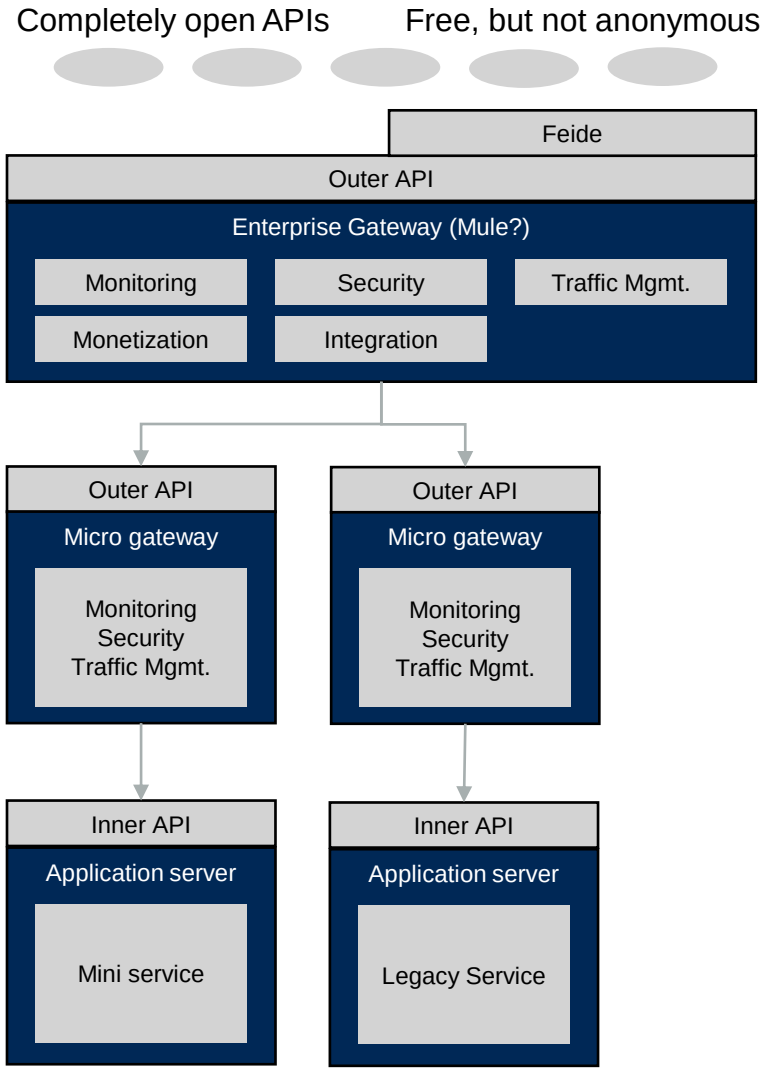




- Gartners Magic Quadrant indicates that MuleSoft (that is already in place) is rated as a top vendor for API Management (NB that Gravitee is not being rated by Gartner
- The assessment of API management should be done based on functional fit, total cost of ownership and availability of skilled resources
- The list to the left indicates critical API management capabilities in a “API integration” setting, which is in line with UNIT’s strategic direction
- MuleSoft scores the highest for this scenario

Gartner

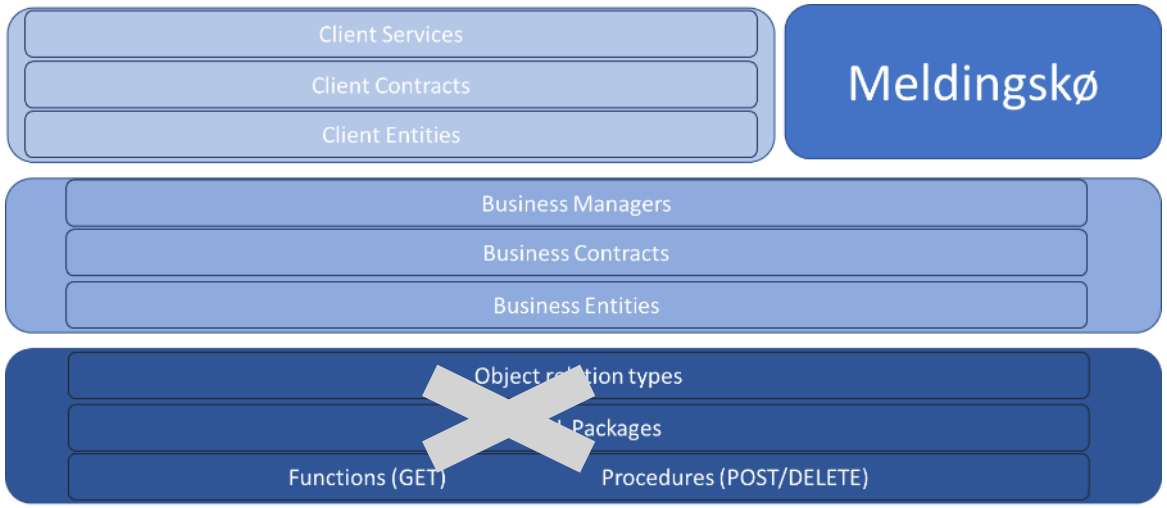
In order to become a platform provider UNIT must leverage the outer/inner API concept and API mediation



- Make a distinction between north/south (from the outside towards the inside of “FS API”) and east/west APIs where services on the “inside” call each other
- North/south mediation applies to interactions between API consumers and published APIs. A published API, also known as an “outer API,” is an API that is designed to be consumed and shared by internal and external consumers (such as mobile, web and conversational apps, internal integration flows, internet-enabled devices, third-party partners, and other applications and services).
- An outer API can also be packaged as a product and become the foundation for a digital product or partner ecosystem.
- A north/south mediation layer maps the outer API to the service implementation’s inner API and enforces runtime policies.
- The inner APIs relies solely on protection from the outer API’s, no service should be able to access another inner API

Gartner

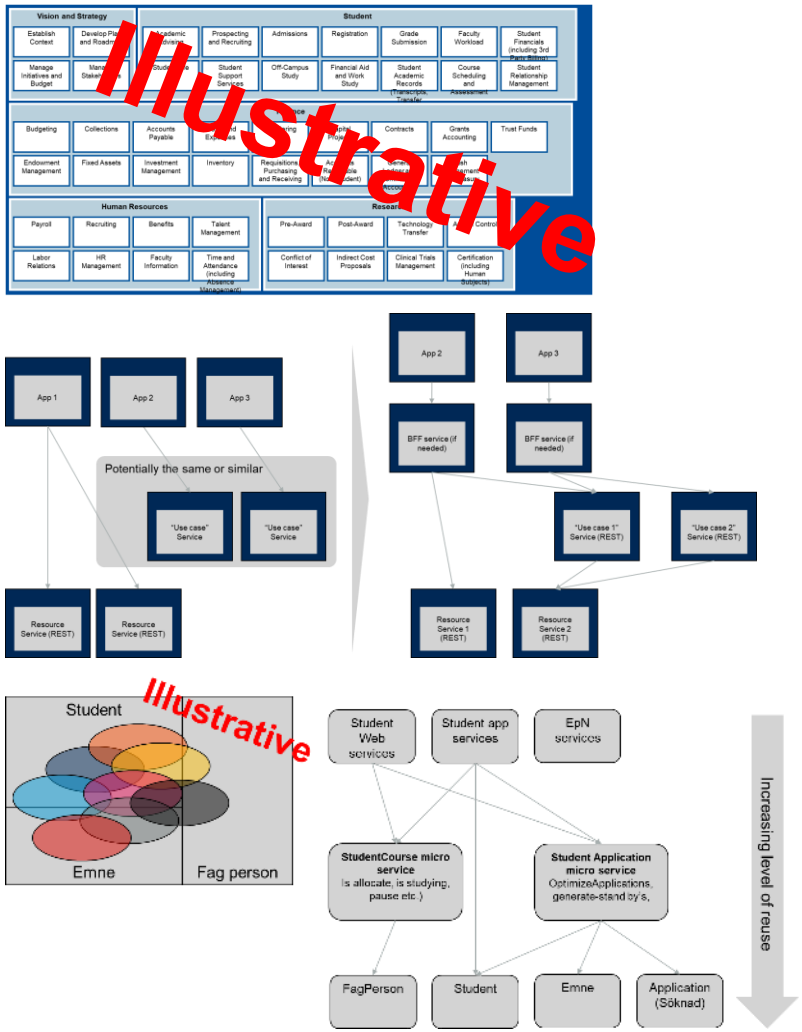
Disentangle the API layer from the database by removing the PL/SQL packages for integration



- The tight coupling to the database and the complexity of the PL/SQL packages that only offers a partial benefit (that layer cannot do all formatting or applying all business logic anyway), as such it needs an additional implementation elsewhere (in the API layer).
- Investigate an alternative approach to access the data than through stored procedures e.g. Object Relationship Mapper
- There is a benefit from decreasing the dependency to Oracle:
  - Decrease dependencies to scarce resources
  - Improve horizontal scalability due to implementing multiple data persistence platforms/instances
  - Improve overall performance by providing the right persistency service for the right type of data
  - Improve overall performance by removing unnecessary processing in the database and move it to the API layer
  - Decreases overall vendor lock in
- Isolate business logic to fewer places especially since the PL/SQL packages won't have the opportunity of resolving all kinds of business logic anyway

Gartner

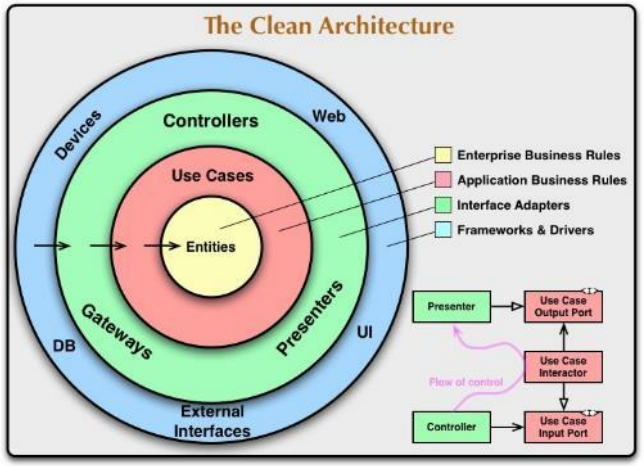
Analyze the functional or resource overlap in the API services based on a capability model (to find use cases or resources)



- Based on a capability model for higher education, identify key domains of resources and actions (e.g. student, staff, institution, education (programs, courses, activities), results etc.
- For each domain map which existing APISs that utilize these resources in order to understand the functional overlap.
- Leverage the design pattern of “back-end for front-end (BFF)” to put channel specific functionalities in that service and strive for more reusable APIs to be designed for reuse and functional stability
- The further back in the API layers the higher level of reusability is expected
- Analyze existing APIs to understand overlap and what consolidations that increases the loosely coupling in the application

Gartner

Consider e.g. clean architecture to create a more overall maintainable solution architecture



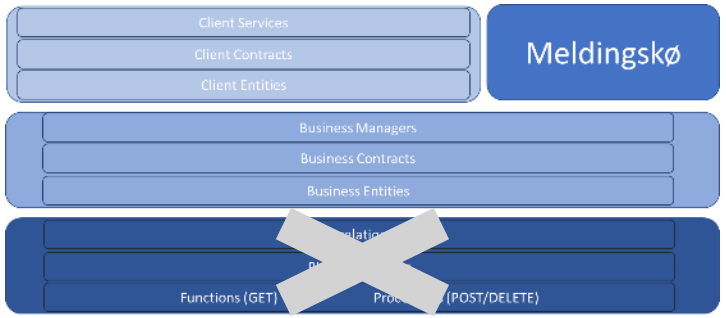
The purpose of a good architecture is to defer decisions, delay decisions. The job of an architect is not to make decisions, the job of an architect is to build a structure that allows decisions to be delayed as long as possible.

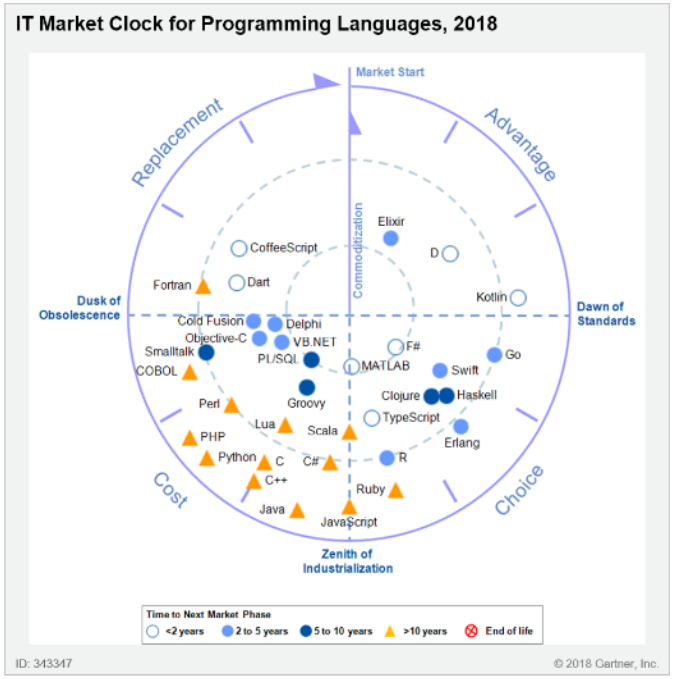
Robert C. Martin (aka Uncle Bob)

- With (more) modern high performing languages with a tighter scope and ambition the solution architecture becomes different and less abstract
- Applying concepts like “clean architecture” (Robert Martin), Hexagonal architecture (Alistair Cockburn) to identify suitable layers and components
- Isolate dependencies to e.g. RDMBS in the proper layer and adhere as much as possible to ANSI SQL and other similar standard objects
- Consider leverage an Object Relational Mapper to minimize the need for SQL in the API layer and don’t create new queries for every kind of scenario. If the database is properly managed row data will be kept close (physically) and selecting all columns vs few may not have the expected outcomes
- Revisit the Java EE decision and evaluate if other technologies are relevant such as Go, Scala etc.

Gartner

Reconsider the long term internal design pattern and disentanglement from the database

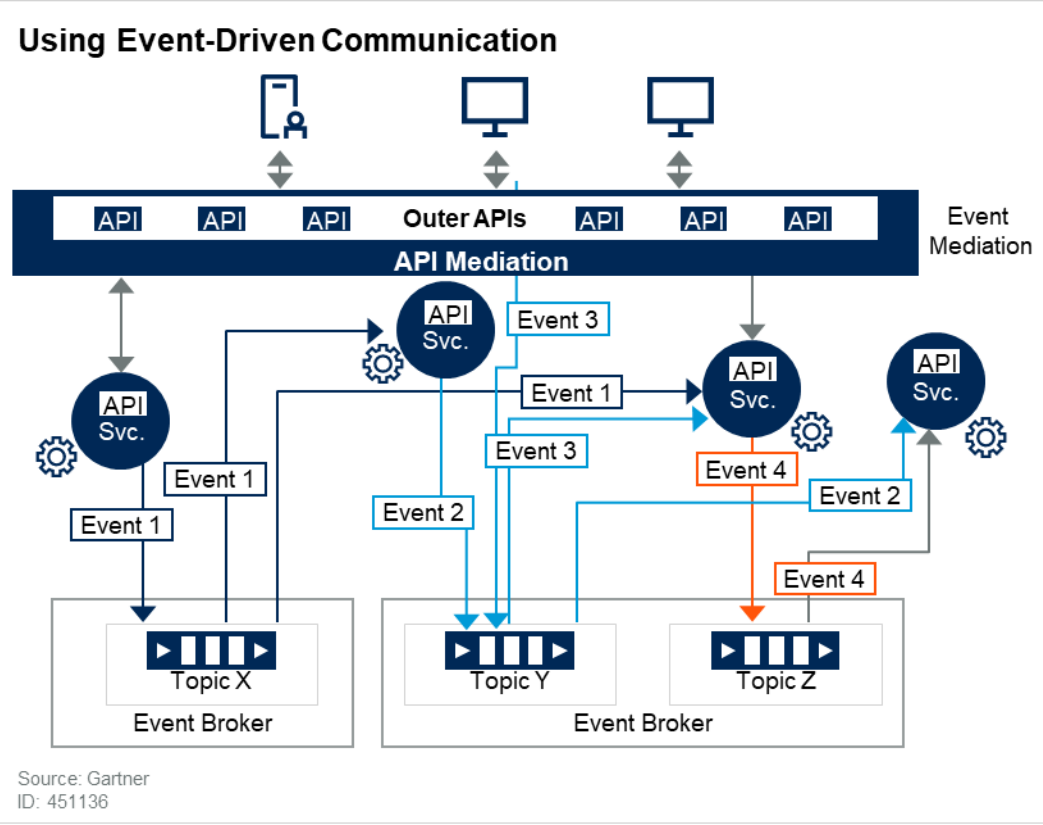




- The current FS API architecture consists of multiple layers and several classes with a distinct purpose, this is well in line with principles of loose coupling and cohesion
- However, this a legacy from Java EE does create a quite complex solution and the level of abstraction is potentially over ambitious
- There is a potential risk of performance issues when there are multiple layers and late binding and dependency injection that during runtime makes all decisions at runtime dynamically rather than compile time
- It can also drive a maintenance effort where lots of classes needs to be changed to deliver a requirement
- Rather than large, complex models with inheritance and the entire business domain in one place we typically see smaller micro-service bundles that ensure proper business rules and persistence in a suitable data model
- Consider other options to Java, (more) emerging technologies like Go, Haskell, Erlang and Scala
  - They offer interesting capabilities such as function orientation. Statically compiled (high performing) language without full object orientation
  - These languages typically drive a new approach and solution architecture to business problems that are more lean and straight to the point

Gartner

Implement event streaming technologies to propagate relevant high level business events and achieve a loosely coupling



Source: Gartner
ID: 451136

Key Findings

- *Event-driven architectures (EDA) use event notifications as a primary mechanism for asynchronously communicating information and triggering application processing.*
- *EDA is a well-defined and long-established architecture paradigm, but this has not yet led to widespread standardization or codification of best practice similar to SOA and REST.*
- *EDA is a natural fit for modern distributed application architectures, including microservices and serverless architecture. It enables minimal coupling and flexible composition of components.*
- *Successful EDA is more dependent upon learning the skills and practices for design and execution than on specific technologies.*

Recommendations

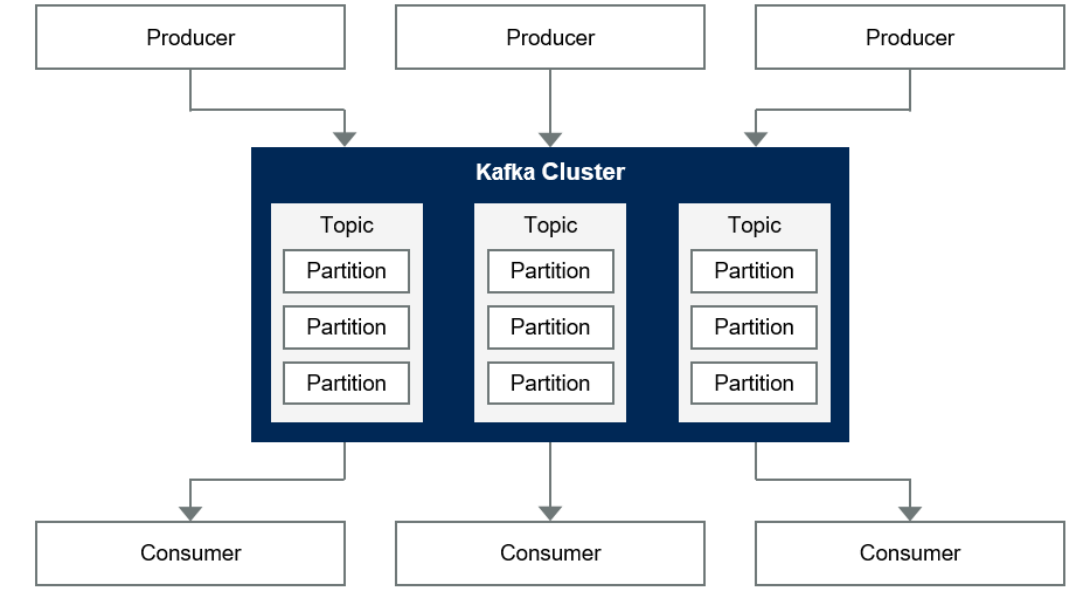
- *Technical professionals responsible for architecture and integration of digital business solutions:*
- *Identify and harness existing pockets of EDA expertise and technology, such as message-oriented middleware, in your organization. Harvest what you find and leverage it across your development, integration and analytics teams.*
- *Always combine EDA with other architecture paradigms, including service orientation and streaming, because the communications in a system will also include requests and commands.*
- *Design and manage event payloads as both programmatic interfaces and data structures. They trigger actions, communicate information, and form the basis for data persistence and analysis.*
- *Focus on developer experience when designing event-based APIs. Use open-standard protocols and publish interface specifications that define event types, payloads, topics and endpoints.*
- *Don't use EDA for communicating and managing states that must be "perfectly consistent." The asynchronous nature of EDA means your system must be able to tolerate eventual consistency"*

Source: Gartner
Applying Event-Driven Architecture to Modern Application Delivery Use Cases, 2019-5-13 | ID: G00377490, Olliffe, Gary

To read about it, Gartner Research: https://www.gartner.com/document/3980382

Gartner

Implement event streaming technologies, such as Apache Kafka

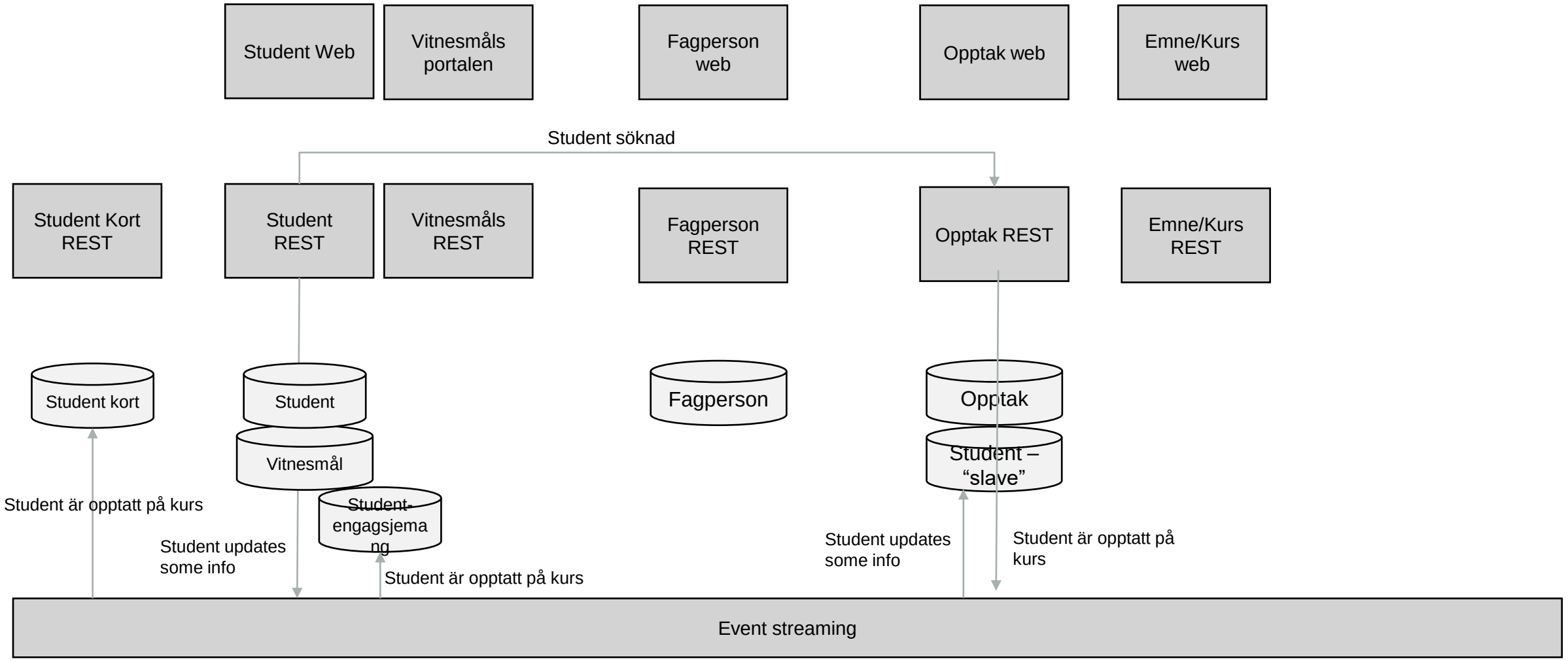


- Assess the need for technical capabilities within the events in place and future needs
- Assess if Apache kafka (which is already in place) is the proper platform and if so, ensure sizing and implementation
- Use event driven architecture to facilitate as loose coupling as possible between cohesive services
- Communicate with distinct high(er) level business events (new student, updated communication profile, admitted to course etc.)
- Avoid synchronous behavior (i.e. that service 1 posts an event and then needs another service to post a follow up event)

To read about it, Gartner Research: https://www.gartner.com/document/3969739

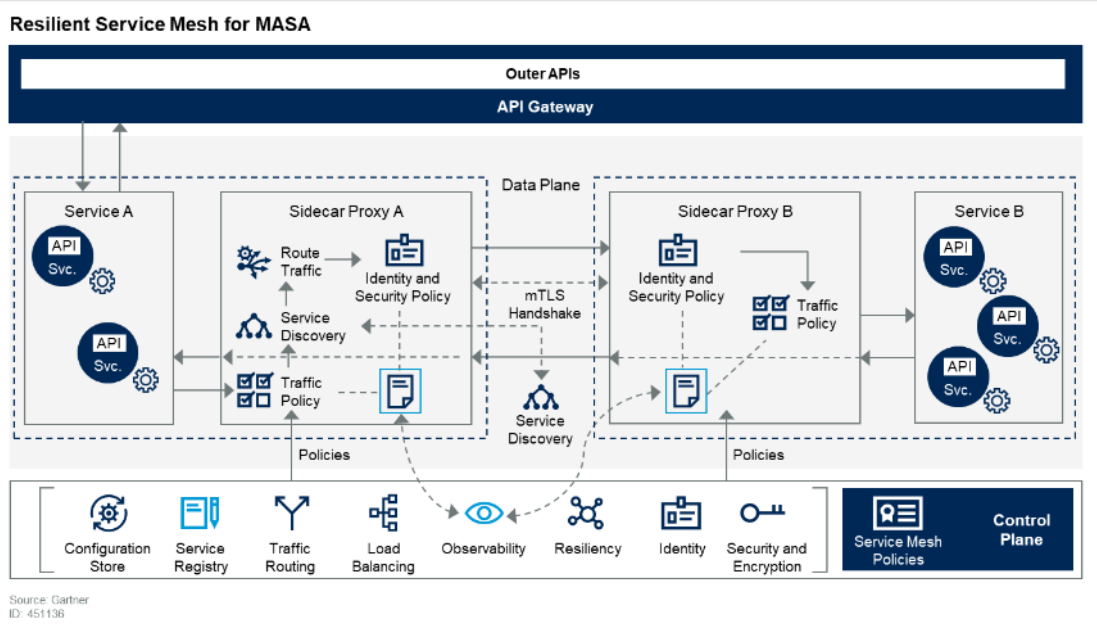
Gartner

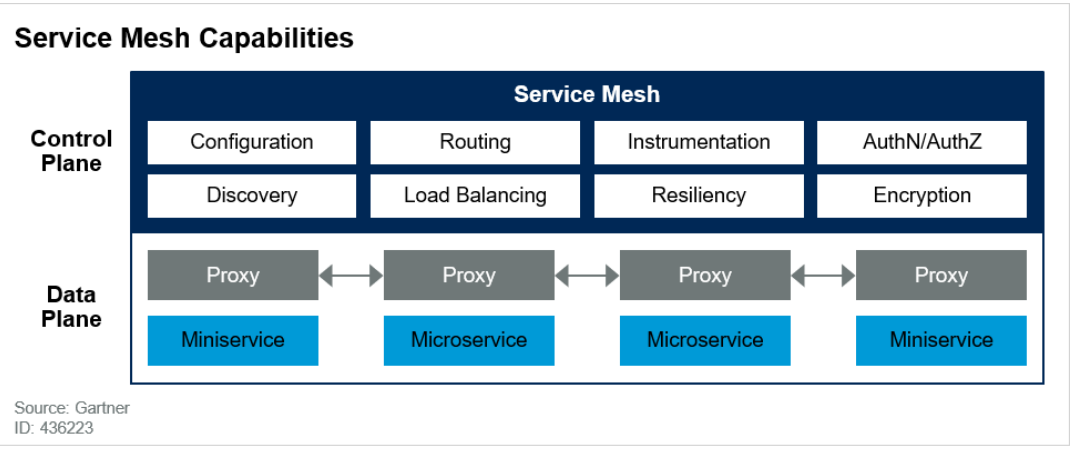
Sample events across potential services



Gartner

Prepare for service mesh and separation of business logic and policies etc.



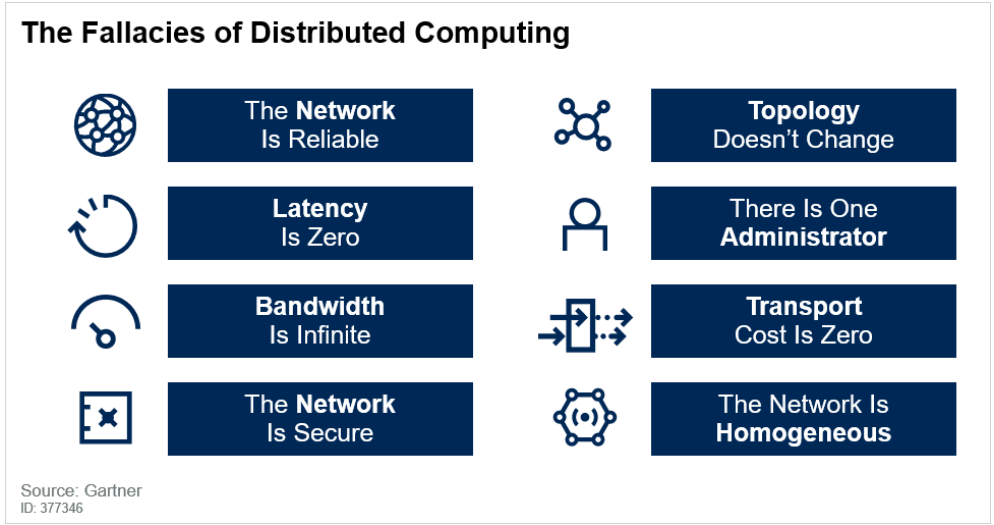


- The more open the architecture becomes, the more important it becomes to protect the back end from load changes and security breaches
- Container approaches using e.g. Docker/Kubernetes are rapidly increasing in popularity and works extremely well with a micro services design
- In order to achieve the proper horizontal scalability a mesh is imperative providing the necessary capabilities:
  - Service discovery
  - Health checking
  - Traffic routing
  - Load balancing
  - Authentication and authorization
  - Observability

To read about it, Gartner Research: https://www.gartner.com/document/3980382

Gartner

The service mesh may sound like API management but should be seen in the light of the infrastructure not being as fault tolerant

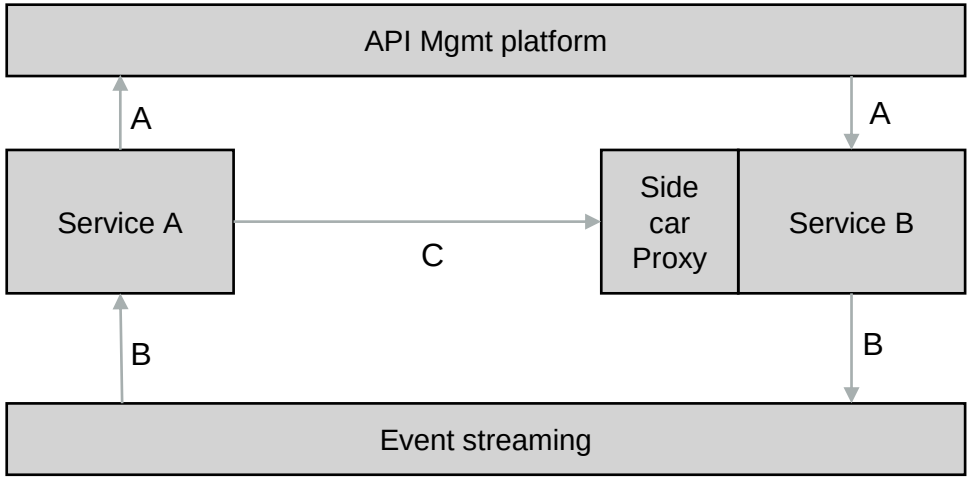


Fallacy	Service mesh functionality
▪ The network is reliable ▪ Latency is zero ▪ Bandwidth is infinite ▪ Transport cost is zero	▪ Configurable resiliency patterns to deal with failures using retries, load balancing and circuit breakers ▪ Additional error handling patterns to deal with network outages and variations in network reliability and capacity
▪ The network is secure	▪ Systematically enforced security policies across all services within the service mesh
▪ Topology doesn't change	▪ Dynamically adjusts network to changing topology and application requirements
▪ There is one administrator	▪ API-driven access to the control plane to allow developers and administrators to instantly see the entire structure of the service mesh and dynamically reconfigure as needed
▪ The network is homogeneous	▪ Abstraction of the application from L3/L4 and any specific implementation details

To read about it, Gartner Research: https://www.gartner.com/document/3907117

Gartner

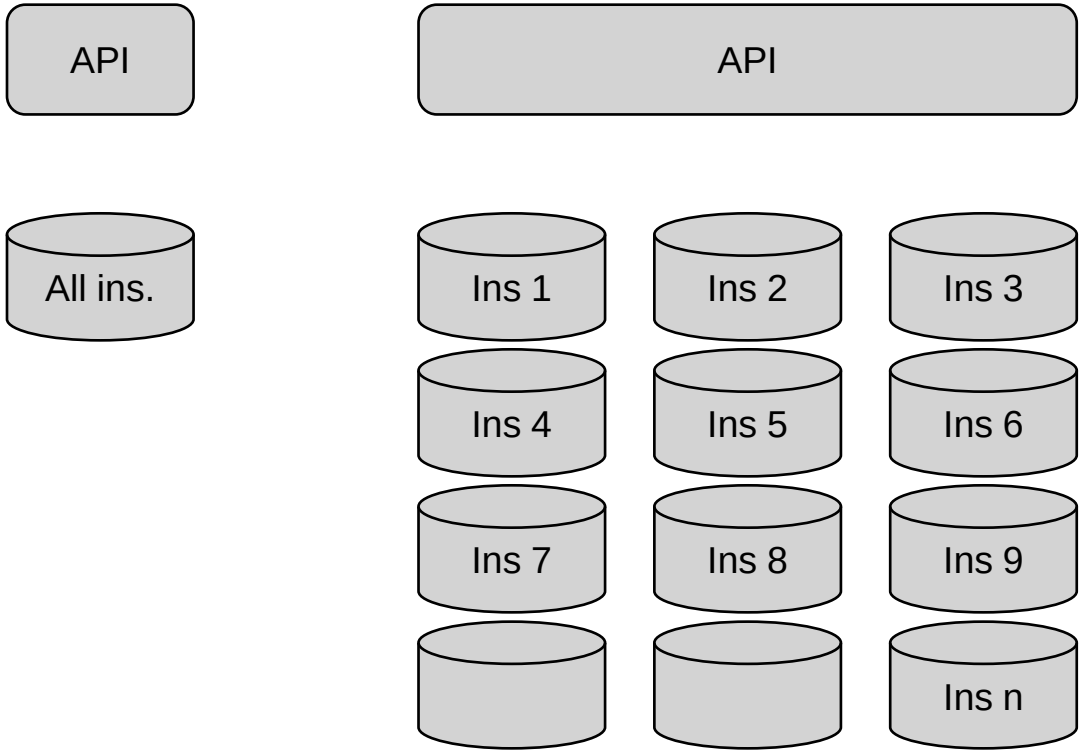
Consider the service mesh to boost performance for service-to-service calls to not create latency or unnecessary distributed data



- An issue with strict interpretation of micro services principles can force a service to only access another services data through an API, (illustrated with A)
- This can impact latency and performance, and an appealing design pattern would be to leverage event streaming and replicate the data (illustrated with B). One consequence with this is it drives additional implementation of data persistence and subscription to events. Cross domain this can be fine, e.g. sending a subset of student data from the student service to the ”course mgmt. service”
- To not drive too much of additional implementation UNIT should consider a service mesh approach with alternative protocols, such as gRPC to create high speed interfaces in a more ”trusted” setting (the mesh is more trusted than the APIs consumers) (illustrated with C)

Gartner

Overhaul the “semi-multi tenant” DB model and the usage of VPD



- Move away from the somewhat odd semi.multi instance + VPD that only gives fundamental data integrity (but not a more contextual data access policy) and it will have a negative impact of performance due to:
  - Additional triggers to be executed to set context columns so data can be filtered afterwards
  - Additional filtering during any query to filter away the other institutions
  - Creates somewhat strange dependencies between the 3-5+ institutions that share the same database instance
- VPD also doesn’t solve more complex context depending access scenarios
- Consider either – full multi tenancy or one institution per instance (recommended) and allow the API for any role based data access control
- In the one institution per instance all VPD capabilities are automatically covered

Gartner

3.4 Relevante vedlegg til funksjonell vurdering

Detaljer fra intervju med studiedirektører

Introduksjon

Som en del av den funksjonelle vurderingen av Felles Studentsystem (FS) ble det gjennomført intervjuer med utvalgte studiedirektører.

Intervjuene hadde som formål å få innblikk i institusjonenes strategiske behov knyttet til studieadministrativt område og funksjonene i FS, samt overordnet strategisk synspunkt på videreutvikling av FS. Intervjuspørsmålene inkluderte;

- Hvor godt understøtter FS de studieadministrative behovene i dag?
- Er det noen fremtidige ambisjoner for studieadministrativt område som kan ha implikasjoner for FS?
- Hvilke behov bør FS dekke i fremtiden, som ikke dekkes i dag?
- Hva er viktige suksesskriterier i videreutvikling av FS?

Det ble gjennomført ett intervju per institusjon som ønsket deltagelse. Studiedirektør var hovedinteressent, men kunne koordinere internt for å supplere egen kjennskap til FS.

Oppsummeringen er basert på gjennomførte intervjuer med følgende institusjoner:

- Norges miljø- og biovitenskapelig universitet (NMB)
- Universitetet i Oslo (UiO)
- Universitetet i Bergen (UiB)
- Lovisenberg diakonale høgskole (LDH)
- Høgskolen i Østfold (HiØ)
- Høgskulen i Volda (HiVolda)
- Høgskulen i Vestlandet (HVL)
- Høgskolen i Molde (HiMolde)
- Universitetet i Sørøst-Norge (USN)
- Norges musikkhøgskole (NMH)
- NLA Høgskolen (skriftlig innspill)
- Vitenskapelige Høgskole (VID)
- Universitetet i Agder (UiA)

FS dekker mer enn godt grunnleggende studieadministrative behov i dag

1

FS dekker mer enn godt grunnleggende studieadministrative behov og funksjoner i dag, men måten funksjonene tas i bruk gjenspeiler ikke hvordan et moderne universitet/høgskole jobber i dag. Dette medfører stor grad av manuelt arbeid og omstendelige arbeidsoppgaver. Det er ytret usikkerhet knyttet til hvor godt FS klarer å dekke komplekse behov/arbeidsflyt og det er gitt uttrykk for at videreutvikling må prioriteres.

Hvor godt understøtter FS dagens behov?

- **Støtter grunnleggende behov på studieområdet:** FS dekker de aller fleste grunnleggende studieadministrative behov (f.eks. godkjenning, opptak, kvalifikasjoner, rapportering, m.m.) men det er allikevel en utfordring i måten FS understøtter behovene.[1]
- **Ivaretagelse av behovsendringer**: Generelt er inntrykket at videreutvikling av funksjonelle områder skjer for seint. Det medfører derfor en usikkerhet knyttet til FS sin evne til å ivareta det som kommer fortløpende.
- **Økende forventninger fra studenter og ansatte:** Selv om de kritiske funksjonene er dekket av FS, holder ikke i takt med måten funksjonene tas i bruk på et moderne universitet/høgskole. For å bedre understøtte behov er det et ønske om forenklet samhandling mellom applikasjoner både i FS-porteføljen og i den helhetlige studieadministrative porteføljen (inkl. f.eks. Canvas).

Hvor viktig er FS for institusjonen?

- **FS er virksomhetskritisk og differensierende for norsk UH-sektor:**
  - FS er det mest virksomhetskritiske systemet for studieadministrativt område.
  - FS differensierer norsk UH-sektor fra høyere utdanning i utlandet. Arbeidsprosesser som ellers hadde vært svært krevende er forenklet pga. tilgangen til FS.
  - Kunnskapsdepartementet (KD) har vært tydelig på at det er et stort potensiale i sektoren for ytterligere koordinering og investering i fellesløsninger. Institusjonene ser FS som en viktig brikke for å få til bedre samhandling på tvers av sektoren.

«FS er et nav for det aller meste [på studieadministrativt område], jeg ser ikke at det kommer til å bli noe mindre strategisk viktig».- studiedirektør

[1] Utfordringen detaljeres i påfølgende lysark

FS bærer preg av «strøm på papir» med lav brukervennlighet

- «Brukervennlighet må læres, det er ikke intuitivt» - intervjudeltager

2

FS bærer preg av en pre-digital hverdag med mye manuell kontroll og papir. Det er gitt eksempler der lav brukervennlighet og komplekst brukergrensesnitt har medført kritiske feil med konsekvenser for institusjonene og for studentene. Tungvinte arbeidsoppgaver og saksbehandling fører også til mye dobbelt arbeid og en ustrukturert arbeidsflyt. FS oppleves som uoversiktlig og det er et ønske om å gjøre systemet mer integrert og enhetlig.

Utfordringer med dagens FS-system:

- **Komplekst og gammeldags arbeidsflate:** Modulene oppleves som låste med lite samspill i mellom. Mye manuell flytting og kopiering av data mellom moduler og øvrige applikasjoner i studieadministrativt porteføljе. FS gjenspeiler ikke måten personer jobber i dag med behov for mer integrerte arbeidsprosesser. FS krever dyptgående kunnskap for å tas i bruk riktig.
- **Vanskelige integrasjoner**: Størst irritasjonsmoment hos institusjonene er knyttet til interne integrasjoner. FS integrasjoner er ikke «*moderne nok*» for å ivareta behov for integrasjoner med «*moderne systemer*», eksempelvis inkluderer dette Canvas, Leganto og ePhorte. Flere løser integrasjonsproblematikken ved å bruke ekstern leverandør for å bygge integrasjon.
- **Manglende kontrollfunksjoner:** Som følge av manglende kontrollfunksjoner er det vanskelig å identifisere og håndtere avvik, noe som fører til mye ekstraarbeid og en rekke utilsiktede feil med potensielt store konsekvenser for studenter og institusjoner.
- **Store konsekvenser ved feil:** Kompleksiteten i brukergrensesnittet og avhengigheten av manuelt arbeid fører til menneskelige feil, økt pengebruk, flere årsverk og potensielt avvik i studenters studieløp. De store institusjonene bruker flere årsverk for å gjøre revisjoner og kvalitetssikring. De mindre institusjonene er sårbare pga. manglende lokal infrastruktur rundt studieadministrasjonen.
- **Tungvinte arbeidsoppgaver og saksbehandling:** Det oppleves at FS er utviklet til 70-80% av sitt potensialet, noe som ofte medfører at FS er tilrettelagt for behovene men ikke for arbeidsflyten og de faktiske aktivitetene. F.eks. STAR rapport funksjonen er bra, men tilgjengeliggjøring av rapportene til beslutningstakere oppleves som krevende og manuelt.
- **Applikasjoner som ikke tas i bruk**: Opplever at det kommer mange nye applikasjoner og tjenester som er blitt nyutviklet men ikke nødvendigvis gode nok til at institusjonene ser verdi av å ta de i bruk. Applikasjonene EpN, FLYT, GAUS og Nomination ble gitt som eksempler.
- **Ikke tilstrekkelig tilpasset faglig virksomhet:** Mange fagpersoner ser ikke nytten eller verdien av f.eks. Fagpersonweb eller EpN pga. lite tilpasning iht. deres rolle.

«Det har skjedd veldig mange endringer på studieadministrativt område de siste årene, systemet [FS] har ikke utviklet seg raskt nok for å ivareta disse.»
- intervjudeltager

Fremtiden vil by på flere integrerte arbeidsprosesser og et mer komplekst økosystem

3

FS er dekkende for studieadministrativt område i dag, men når det gjelder fremtiden og ambisjonsnivået beskrevet i sektorens- og institusjonenes strategier er det utfordringer som ikke er løst og muligheter som ikke er dekket av dagens plattform. Ønsker for fremtiden inkl. bedre samspill og deling mellom institusjoner, mer selvbetjening og automatisering, mer personlig saksbehandling, samt bedre støtte for internasjonalisering, livslang læring og læringsanalyse.

Fremtidige ambisjoner som kan ha/allerede har implikasjoner for FS:

- **Samhandling med økosystemet (lokalt, nasjonalt og internasjonalt)**: Det medfører behov for sammenhengende tjenester, mulighet for gjenbruk av informasjon på tvers av applikasjoner, institusjonsgrenser og landegrenser.
- **Svekkende grenser mellom institusjoner og økt samarbeid**: Dette gjenspeiles i økende behov for samhandling, at student/ansatt kan forekomme i samme person, bruk av sensurer fra andre institusjoner, flere større institusjoner pga. sammenslåening og dermed flere fler-campus miljøer.
- **Mer automatikk/selvbetjening i arbeidsprosesser**: Det er behov for standardisering, forbedring og effektivisering, samt enkel og automatisk overføring av data mellom applikasjoner. Tilrettelegge for mest mulig selvbetjent og enkelt grensesnitt for at fagpersoner og studenter kan gjøre mest mulig selv.
- **Brukertilpasset og personlig brukerinteraksjon og saksbehandling**: Det medfører behov for at data i FS kan kobles til «kunnskap» om studenter og studier for å gi et mer tilpasset grunnlag for saksbehandling og beslutninger. Institusjonene ser behov for et tjenestedesign perspektiv med bedre integrasjoner og flyt mellom alle systemer på studieadministrativt område.

Transformasjonsdrivende trender for UH-sektor ble også omtalt:

- **Internasjonalisering og mobilitet**: FS oppleves som et hinder for økt internasjonalisering og mobilitet. Samarbeid med gode miljøer både nasjonalt og internasjonalt påpekes som svært viktig og en forutsetning for å fremme norske forsknings- og utdanningsmiljøer. Dette er tett koblet til opptak, veiledning, søknadsbehandling, nominering og forvaltning av utvekslingsavtaler.
- **Læringskraftig utvikling og kompetansereformer:** Satsningen på blant annet livslang læring betyr at UH-sektor ser færre «tradisjonelle studenter». Modulbasert læring med persontilpassing, en fleksibel semester tilnærming (opptak, veiledning og eksamen) er fremtiden. Samarbeid med næringslivet står sentralt her og bør støttes av FS.
- **Analytics og beslutningsstøtte**: FS er den viktigste datakilden for studieadministrativt område, men pga. et tungvint grensesnitt, krevende tilgjengeliggjøring og lite dynamisk rapportering er det vanskelig å snu data til innsikt. Behov for kontinuerlig og automatisk analyse av data som vil gi mulighet for prediktive analyser som kan f.eks. bistå med tidlig oppdagelse av frafall. Behovet for innsikt og beslutningsstøtte er økende og det er et stort potensialet som ikke er utnyttet i dag.

Gartner

FS suksess er avhengig av bl.a. tempo i videreutvikling, økt brukermedvirkning, samt gode og tydelige prioriteringer

4

Sektor har ytret suksesskriterier knyttet til funksjonell og teknisk forvaltning i tillegg til samhandling og styring. Suksesskriterier for å bedre støtte moderne arbeidsprosesser er økt samhandling med deling på tvers av institusjonsgrenser, et raskere utviklingstempo, forenklet brukergrensesnitt, mer sømløst dataflyt, forbedre dataintegritet, bedre utnyttelse av sektorkompetanse, forutsigbar finansiering, m.m.

Funksjonelle og tekniske suksesskriterier:

- **Høyere tempo på videreutvikling:** Organiseringen og styringen tilknyttet FS oppleves som lite fleksibel. For å lykkes må videreutvikling skje i samme tempo som endringene innenfor studieadministrativt område. Det er viktig at politikk ikke går på bekostning av effektiv forvaltning og nødvendig videreutvikling.
- **Forenkle brukergrensesnittet:** Moderne arbeidsflate og grensesnitt som er intuitivt og tilpasset et moderne universitet/høgskole. Arbeidsflaten bør være intuitiv og enkel med gode integrasjoner som gjør at den forenkler og effektiviserer arbeidsprosessene til de ulike brukergruppene.
- **«En vei inn» med fellesplattform:** En helhetlig tekning rundt FS, IAM og Arkiv vurderes som suksesskriterie for alle tre pågående prosjekter på nasjonalt nivå.
- **Sømløst dataflyt og bedre integrasjoner:** Unngå mest mulig manuelt og duplikat arbeid ved å tilrettelegge for mer sømløst dataflyt mellom moduler og øvrige applikasjoner som f.eks. Canvas, Arkiv og Leganto. Tilrettelegge for mer sammenhengende og digitalisert saksflyt, studieadministrative prosesser og rutiner.
- **Mulighet for lokale tilpasninger:** Det er varierende behov i sektoren ut i fra type institusjon, størrelse, antall campus, internasjonalt samarbeid, type bruker, m.m. Med mulighet for lokal tilpasning så kan saksbehandlingen oppleves mindre omstendelig noe som kan føre til økt effektivitet i saksflyten.
- **Forutsigbar opplæring og tilgang til support:** For at institusjonene skal bruke FS riktig, forstå mulighetsrommet og få mest mulig verdi av FS funksjonene bør det utvikles/oppdateres instrukser, brukermanualer, nettbasert og personlige kurs på nasjonalt nivå. Tilgang til mer utvidet brukerstøtte og opplæring fra UNIT.
- **Bedre dataintegritet:** Manglende kontrollmekanismer og kvalitetssikring av data som blir tastet inn i FS kan føre til store konsekvenser for både institusjon og student, som f.eks. ikke fullført eksamen, manglende vitnemål, m.m. For institusjonen kan dette gi økonomiske konsekvenser og skade omdømme.
- **Muliggjøring og utvikling av tjenester:** Evne til å understøtte digital tjenesteproduksjon for studenter og faglærer. Understøttelsen bør skje i form av å forenkle bruk og tilgjengeliggjøring av data til institusjonenes egen tjenesteproduksjon. Det er viktig at tjenesteproduksjon ikke hemmes av FS.

Suksesskriterier knyttet til samhandling og styring:

- **Utnytte sektorkompetanse:** Innebærer å ha god forståelse for institusjonenes faktiske behov og arbeidsflyt, samt bedre utnyttelse av kompetansen som allerede eksisterer hos institusjonene. Viktig at FS tilpasses behovene til institusjonene og ikke at institusjonene må tilpasse sine behov til FS.
- **Forutsigbar finansiering:** Uten forutsigbar og tilstrekkelig finansiering får sektor en liten forutsigbar leveranse. Opplever i dag at mye blir satt på vent pga. manglende midler som også medfører manglende utviklingskompetanse.
- **Tydelige prioriteringer og økt transparens**: Tydelig og transparente prioriteringer, samt god kommunikasjon er kritisk for å bevare tilliten fra institusjonene. FS-klienten og databasen bør prioriteres.
- **Tettere samhandling på tvers av sektor:** Bedre samarbeid på tvers av sektor vil gi aktivitetene og prioriteringene mer legitimitet. Mindre til mellomstore institusjoner nevner at legitimiteten svekkes når institusjoner velger å gå sin egen vei. «*Nasjonal dugnad er helt kritisk*».

Gartner

Studiedirektørene beskriver tre kjennetegn på hva FS er i fremtiden

5

Studiedirektørene er svært opptatt av at sektoren opparbeider seg et enhetlig syn over alle studentene i Norge og mener at FS er kritisk for å få dette til. De fleste ser for seg en fremtid der FS er en kjerne for nasjonal studentdata, en autoritativ kilde som muliggjør deling på tvers av institusjonsgrenser. FS bør være utviklet for et moderne universitet med gode moderne integrasjoner. Det er viktig at FS i fremtiden videreutvikles etter sluttbrukerbehov istedenfor at behov må tilpasses FS.

Kjennetegn på hva FS er i fremtiden:

1. **En kjerne for nasjonal studentdata**
   - *«Datanav som understøtter hele utdanningsløp [inkl. etter- og videreutdanning]»*
   - *«Masterdatasystem med gode APIer»*
   - *«Nasjonal database som muliggjør deling på tvers av institusjonsgrenser»*
   - *«En kjerne av studentdata»*
   - *«Videreutviklet for å håndtere sensitive data»*
   - *«FS som autoritative kilder på det meste»*
2. **FS bør være tilrettelagt for gode integrasjoner:**
   - *«Tilrettelegge for mer lokal tilpasning av omkringliggende applikasjoner»*
   - *«Studieadministrative prosesser og andre ting kan dekkes av lokale systemer»*
   - *«Utviklet for et moderne universitet med behov for moderne integrasjoner»*
   - *«En smart svart boks»*
3. **FS som et rendyrket og moderne brukergrensesnitt og tjenesteintegrator**
   - *«Tilrettelegger for nasjonal IAM»*
   - *«Et effektivt verktøy for arbeidsstøtte [for saksbehandlere]»*
   - *«Bygget opp etter studieadministrative og sluttbruker behov, ikke motsatt»*
   - *«En tjenesteleverandør som sikrer samkjøring mellom institusjoner»*
   - *«Sentralisert avlastning av lokale behov» (kontroller som sikrer kvalitet av data på tvers av sektor)*
   - *«Kompetanseutvikler»*
   - *«Videreutvikling av en nasjonal styringsportal»*

«Spørsmålet bør egentlig være - hva skal FS som database være?»
-intervjudeltager

Ved forbedring av FS tror intervjudeltagerne at ytterligere effekter kan oppnås som f.eks. økt produktivitet og kostnadsbesparelser

6

Ved videreutvikling av FS antar studiedirektørene at det vil kunne utløse betydelige gevinster når det gjelder effektivitet, kvalitet og strategisk måloppnåelse. Alle institusjonene må sammen gripe muligheten for å sikre at FS videreutvikles iht. dagens og fremtidens behov og dermed sikre bedre kollektiv utnyttelse av effektene. Det forventes at gevinstene vil bidra til effekter som f.eks. økt produktivitet av studieadministrative ansatte, økt kvalitet av studieløp, forbedret fellesskapsfølelse på sektornivå, m.m.

Gevinstområde	Effekter
1. Effektivitet	• Økt produktivitet • Redusert ressursbehov for studieadministrative oppgaver • Brukervennlighet • Bedre arbeidsmiljø for studieadministrative ansatte • Kostnadsbesparelser
2. Kvalitet	• Mindre risiko for avvik og feil • Fremstå mer enhetlig på sektornivå • Økt kvalitet i saksbehandling • Større grad av forutsigbarhet i prioriteringene for videreutvikling • Økt tilfredshet
3. Strategisk viktig	• Bedre understøttelse av strategiske ambisjoner • Økt fellesskapsfølelse på sektornivå • Økt kompetanseoverføring

«Bedre integrasjoner hadde ført til økt effektivitet og innsparing på ressurser.»
-intervjudeltager

«Ved å digitalisere rutiner som saksbehandlere gjør i dag kan vi nok spare ca. 15-20% av tiden til alle studieadministrative ansatte»
-intervjudeltager

Identifiserte behov som ikke dekkes helt eller dekkes bare delvis fra FS

Innspill fra intervju med studiedirektørene

- Plagiat verktøy
- Nettskjema
- Timeplanlegging og oppmøteregistrering
- Praksiskoordinering og aktiviteter knyttet til praksisopphold
- Koordinering for bruk av eksterne sensorer
- Analytics kapasitet
- Tilgjengeliggjøring av rapportering til beslutningstakere
- Selvbetjening på fagperson og student nivå
- Personforekomst student/ansatt samtidig
- Gode integrasjoner mot arkiv og canvas
- Semesterbetaling
- Klargjøring av eksamen
- Registrering av f.eks. politiattester
- Internasjonalopptak
- Bedre støtte for «fler-campus miljøer»
- Felles grader
- GDPR henvendelser
- Studieplanlegging

NB! Listen er en blandet liste med både behovsområder og konkrete forslag. Dette er en bruttoliste som går utover det som er tidligere beskrevet i materialet.

Innspill fra Fagutvalg for utdanning (fra møte 1. april 2020)

- Ta hensyn til universell utforming i FS

Utdrag fra brukerundersøkelse for FS-/superbrukere

Detaljer i vedlagt Excel fil til sluttrapporten:
Unit - Vurdering av FS - Anonymisert - Brukerundersøkelse 032020 Superbruker.xlsx

Gartner

FS-/superbruker – respondentoversikt

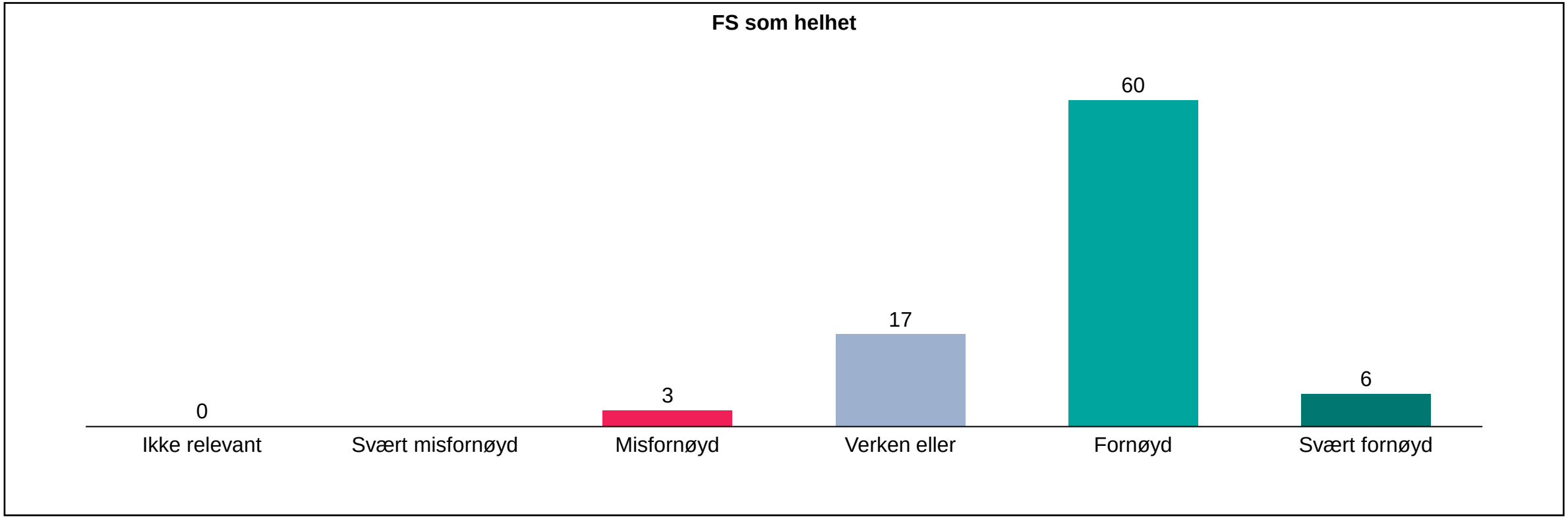
94 fra følgende institusjoner mottok brukerundersøkelsen

Institusjon
Forsvarets høgskole
Høgskolen i Molde
Høgskolen i Østfold
Høgskulen i Volda
Høgskulen på Vestlandet
Høyskolen Kristiania
Kriminalomsorgens høgskole og utdanningssenter KRUS
Lovisenberg diakonale høgskole
MF vitenskapelig høyskole for teologi, religion og samfunn
Nord universitet
Norges miljø- og biovitenskapelige universitet
Norges teknisk-naturvitenskapelige universitet
OsloMet – Storbyuniversitetet
UiT Norges arktiske universitet
Universitetet i Agder
Universitetet i Bergen
Universitetet i Oslo
Universitetet i Stavanger
Universitetet i Sørøst-Norge
VID vitenskapelige høgskole

86 (91%) besvarte brukerundersøkelsen

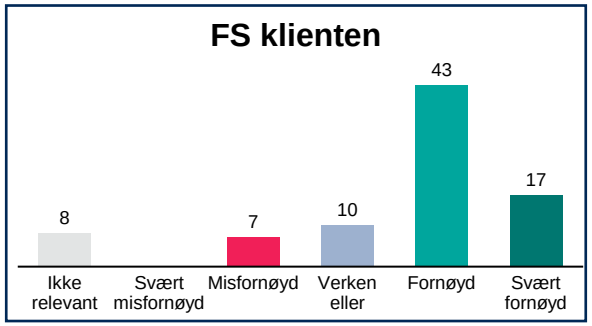
	Antall års erfaring i bruk av Felles Studentsystem (FS) til studieadministrasjon							
Brukergrupper	**0 år**	**1 år**	**2 år**	**3 år**	**4-6 år**	**7-8 år**	**9+ år**	**Delsum**
FS-bruker	1	1	1	1	11	4	21	40
Superbruker			1	1	7	8	29	46
Delsum	**1**	**1**	**2**	**2**	**18**	**12**	**50**	**86**

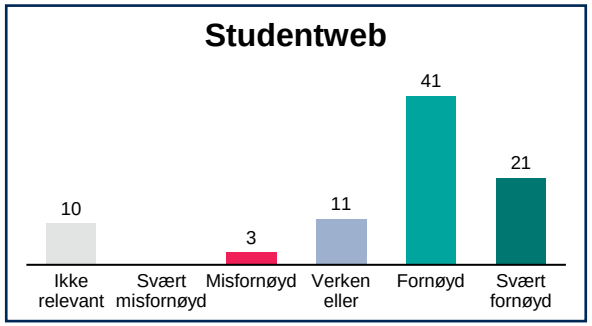
FS-/superbruker - oppsummering av tilfredshet FS som helhet

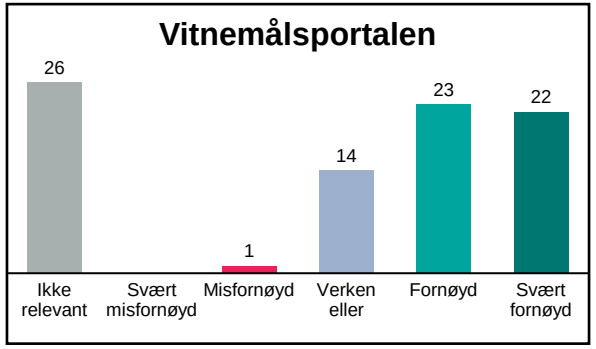


FS-/superbruker - oppsummering av tilfredshets FS applikasjoner (1/4)

Hvor fornøyd er du med applikasjonen…







Kommentarer* fra (svært) misfornøyde…

- *Lite **brukervennlig** og vanskelig å forstå for nye brukere*
- *Ekstremt **lite intuitivt** grupperingssystem for funksjoner*
- *Har **aldri klikket og scrollet** så mye for å få selv den minste ting gjort*
- ***Feilmeldinger** er teknisk rette men **uforståelige** for folk uten teknisk innsikt*

- *Altfor lett at studentene **tror de er semesteregistert uten å være det***
- ***Bildet er vanskelig å lese med mindre du forstår deg på strukturen**. Feltene er ikke navngitte og det forutsetter at bruker kan systemet svært godt*

- ***Eldre utdanninger fremkommer ikke alltid korrekt i Vitnemålsportalen**. Det burde derfor kun vært utdanning etter en gitt dato som ble overført til Vitnemålsportalen.*

Kommentarer* fra verken eller…

- *Bra hvordan det fungerer **men trenger ny front end***
- *Kunne godt gjøres **forbedringer** for å gjøre det **forståelig for nyansatte***
- *FS er **litt rotete og har et brukergrensesnitt som er lite brukervennlig***
- *Mange **registeringer er en tungvint prosess***

- ***Dårlig brukervennlighet** i semesterregistering og Utdanningsplan – studentene gjør mye feil*
- *Ikke tilpasset brukergrensesnitt for mobil og andre nettlesere enn Internet Explorer*

- ***Tungvint at vi ikke kan simulere hvordan ting ser ut for student***
- *Flott tilbud til nåværende og tidligere studenter. Det eneste vi har problemer med er gamle data med dårlig kvalitet eller ikke eksisterende resultater*
- *Det er en del mangler i Vitnemåls-portalen pga. manglende elektroniske registreringer en del år tilbake i tid. Skulle gjerne kunne velge hvilket år det skal starte å vise data fra.*

Kommentarer* fra (svært) fornøyde..

- *Gir svært gode muligheter for å få ut ekstremt mye informasjon og **dekker svært mange behov***
- *Fungerer bra i de fleste tilfeller*
- *FS klienten er rask*
- *Elsker FS klienten men **trenger endringer** i campusfunksjonalitet, betaling, modultekster, roller/brukeradministrasjon, vurdering, feilmeldinger, varselsloggen*

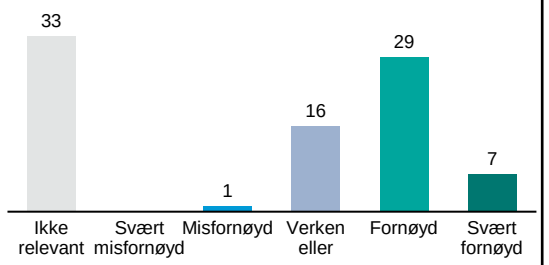
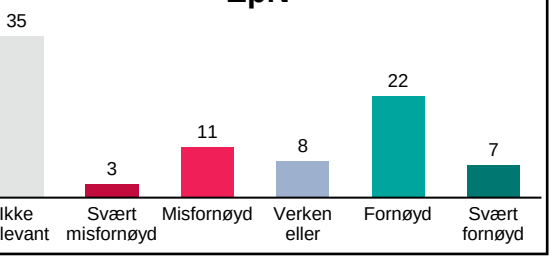
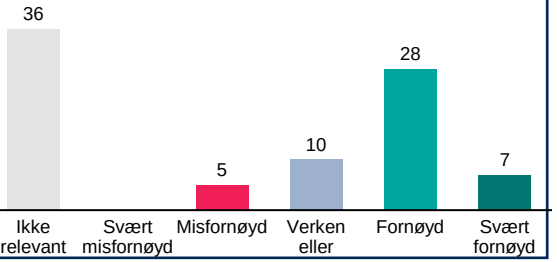
- *Bra applikasjon, men **fungerer dårlig i Internet Explorer***
- *Fungerer bra for de fleste studenter*
- ***Litt tungt å få innført ny funksjonalitet***
- ***Savner** en mer **moderne betalingsløsning for studenter***

- ***Brukervennlig og enkel**.*
- *Veldig bra hvor studenter enkelt kan hente ut karakterutskrift og vitnemål.*
- *Det blir bra når vitnemålet KUN er digitalt*
- ***Fungerer utmerket for nye data**. Litt ugreit for eldre data og måten data har blitt registrert på som ikke dekkes.*

** Utdrag fra kommentarene i undersøkelsen*

Gartner

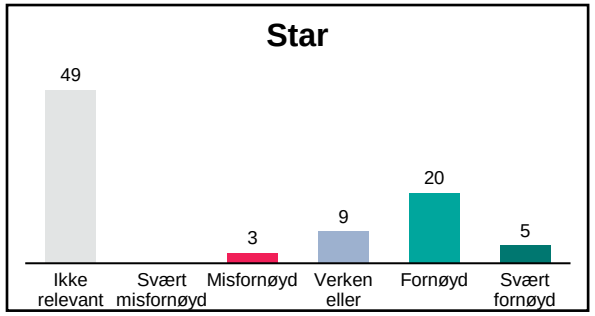
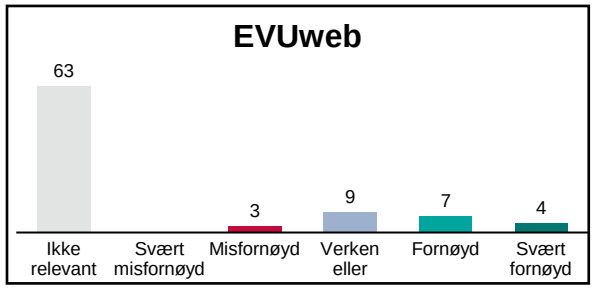
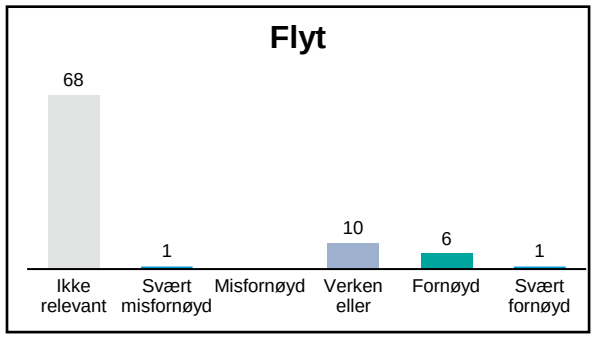
FS-/superbruker - oppsummering av tilfredshets FS applikasjoner (2/4)

Hvor fornøyd er du med applikasjonen…	Kommentarer* fra (svært) misfornøyde…	Kommentarer* fra verken eller…	Kommentarer* fra (svært) fornøyde..
Fagpersonweb 	• *Kan til tider være svært vanskelig å drive brukerstøtte i, og **fremstår mindre intuitiv enn hva Studentweb** gjør. Ønske om en bedre løsning og større integrasjon av digitalt oppmøte.*	• *Vi benytter eksamenssystemet til sensur.* • *«Enda et system faglærer må forholde seg til», faglærer forstår ikke hvorfor de ikke kan bruke Canvas* • *Savner bedre søkefunksjon for fagpersoner, de strever stadig med å finne studenter* • *Brukervennlighet noe dårlig* • *Opplever at sensorer ofte har problemer med å laste opp filer for begrunnelse og de ser ofte ikke forespørslene etter begrunnelse*	• *Har noen grunnleggende mangler som er irriterende og utviklingen går altfor tregt* • *Nyttig og bra verktøy, men mange av de vitenskapelige er ikke klar over funksjonaliteten* • *Ypperlig applikasjon som hjelper faglærere å gjennomføre enkle studieadministrative oppgaver*
EpN 	• *Svært tungvint emneredigering* • *Lite egnet til å holde oversikt over emner* • *Ikke fleksibelt, burde vært tilgjengelig hele tiden for redigering* • *Må jobbe i både FS og EpN for emneredigering* • *Manglende automatisk synkronisering til FS*	• Ikke helt effektivitet i saksflyten • Vurderingsordninger er komplisert • Begrensede integrasjoner med FS. Må jobbe både i FS og EpN samtidig for å utføre oppgaver. • Utfordringer med tekniske feil/feilretting • Mangler mye funksjonalitet	• *Emnerevisjon gjøres et sted* • *Ryddig og enkel* • *Takket være EpN har vi nå kontroll på emneporteføljen* • *Stort sett fornøyd men skjønner ikke studieprogramdelen* • *Savner en del rapporter og en god løsning for programplaner*
Søknadsweb 	• *Blitt verre etter oppdateringene, hatt flere spørsmål om opptak på lenge* • *Mangler å kreve fullstendig søknad før en søker får lov til å levere. Store mengder saksbehandlingstid skusles bort på ufullstendige søknader* • *Mange spørsmål for å veilede studenter hva de skal søke på, ikke logisk hva de skal søke*	• *Fungerer stort sett som det skal men burde vært enklere å søke i studier* • *Har blitt bedre men egner seg ikke for internasjonale søkere* • *Veldig dårlig nynorsk i kvitteringstekster* • *Bør kunne brukes til opptak i PhD*	• *Litt tungvint å få innført ny funksjonalitet* • *Brukervennlig* • *Søknadsprosessen er blitt heldigital, veldig behagelig* • *Fungerer bra* • *Gammel dokumentasjon fra tidligere opptak er inne i ny søknad. Det er uheldig og gjør at vi må etterspørre ny dokumentasjon*

** Utdrag fra kommentarene i undersøkelsen*

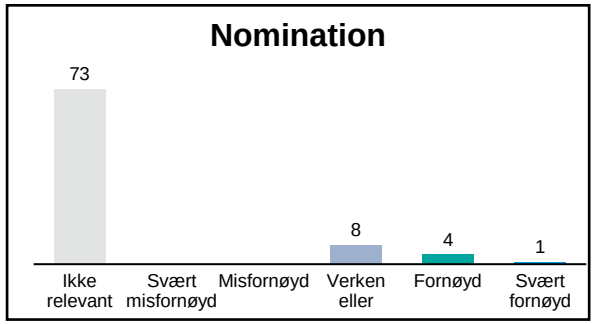
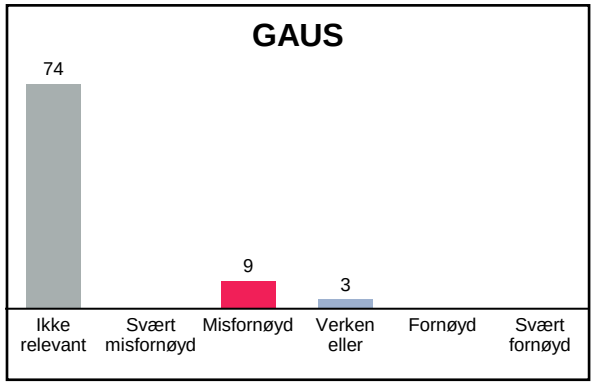
Gartner

FS-/superbruker - oppsummering av tilfredshets FS applikasjoner (3/4)

Hvor fornøyd er du med applikasjonen…	Kommentarer* fra (svært) misfornøyde…	Kommentarer* fra verken eller…	Kommentarer* fra (svært) fornøyde..
	• *Alt for komplisert* • *Vanskelig å finne fram for en som ikke har inngående kjennskap* • *Uoversiktlig, mange rapporter*	• *Gir mange muligheter, men er krevende å forholde seg til* • *Må jobbe med å gjøre data tilgjengelig gjennom bedre løsninger for innlogging* • *Noen uttrekk i dette verktøyet savnes i FS rapportene* • *Tenkt brukt av alle, men vår erfaring er at den er for eksperter med stor kunnskap om statistikk*	• *Gjort det mye enklere å finne fram til en del tall* • *Fint å kunne hente ut data men tungvint å tilpasse til eget behov* • *Litt vanskelig å forstå bruken ved sjelden bruk*
	• *Ganske uoversiktlig* • *Oppsettet og bruken er lite intuitivt og fortsatt for komplisert*	• *Denne er sårt tiltrengt men mangler en del når det kommer til foretaksregisteret så det har ikke blitt benyttet* • *Utfordring med pålogging for ut. Søkere*	• *Godt brukergrensesnitt med mye funksjonalitet* • *Brukervennlig, veldig enkelt å bruke for søker* • *Minus at hvem som helst kan logge seg på med Facebook etc. – skaper mye ekstrajobb* • *Behandling av søknader er komplisert og søker kan ikke foreta prioritering av emner som skaper noe problemer for saksbehandler*
	• *En svært tungvint applikasjon hvor systemene ikke snakker godt nok sammen*	• *Brukervennlig men tungvint å sette i gang pga. problemer med overføring til P360* • *Ikke all overføring til arkivsystem som virker*	• *Veldig arbeidsbesparende* • *Gir mulighet for effektiv samhandling om saker* • *Intuitivt og godt grensesnitt* • *Løsningen fungerer utmerket men har ikke tatt i bruk fordi flyt ikke er integrert mot andre arkivsystem enn Public360*

** Utdrag fra kommentarene i undersøkelsen*

FS-/superbruker - oppsummering av tilfredshets FS applikasjoner (4/4)

Hvor fornøyd er du med applikasjonen…	Kommentarer* fra (svært) misfornøyde…	Kommentarer* fra verken eller…	Kommentarer* fra (svært) fornøyde..
	Ingen kommentarer	• *Bra at samarbeidsinstitusjonene kan samarbeide, men komplekst å få alle elementer på tvers av opptak- og utvekslingsmodulen riktig definert* • *Brukerdokumentasjonen kunne vært bedre* • *Burde vært mulig å nominere til flere opptakstyper* • *Fungerer etter hensikten*	• *Lettet nominasjonsprosessen betraktelig* • *Savner å kunne definere egne forretningsregler* • *Ser bare resultatene av appen. Brukere tuller ofte med fødselsdato og navn* • *Plattform for nominasjon og søknader på en sikker måte* • *Systemet krever veldig mye oppdateringer av kontaktinformasjon*
	Ingen kommentarer	• *Greit med oversikt men ikke hørt om andre som bruker denne enn meg* • *Har lite erfaring med modulen* • *Mange hadde problemer med innlogging for noen år siden* • *Greit med en oversikt men har kanskje utspilt sin rolle nå som resultatutveksling er på plass*	Ingen kommentarer

** Utdrag fra kommentarene i undersøkelsen*

Gartner

FS-/superbruker –forslag til forbedringer

Har du forslag til hvordan applikasjonene kan forbedres til utførelse av studieadministrative oppgaver?

- **Flest forslag i følgende applikasjoner knyttet til:**
  - Brukervennlighet i FS klienten
  - Bedre prosesstøtte / funksjonalitet og brukervennlighet i EpN
  - Forbedring av brukervennlighet i Studentweb
- **Flest forslag går ut på forbedring av brukervennlighet**
  - Mer intuitiv navigasjon
  - Mer forståelige feilmeldinger
  - Flere hjelpetekster.
  - Mer effektiv arbeidsutførelse gjennom prosessoptimalisering, integrasjoner, automatiseringer
- **Mer funksjonalitet for bedre prosesstøtte:**
  - Enkelte oppgaver kan ikke utføres optimalt i FS pga manglende funksjonalitet. Mange forslag går derfor ut på legge til funksjonalitet (småjusteringer).
- **Mer tilgang til data/informasjon**
  - Flere/bedre/mer tilgjengelige rapporter
  - Mer integrerte systemer
- **Bedre brukergrensesnitt**
  - Mer moderne og intuitiv design – oppdatere farger, symboler, figurer, skriftstørrelse
  - Brukergrensesnitt tilpasset mobil og flere nettlesere

- **FS klienten – bedre brukervennlighet og bedre prosesstøtte:** Det bør bli enklere/mer intuitivt å utføre oppgaver. Dette inkluderer mer forståelige feilmeldinger enklere filtreringer, gjøre det enklere å finne fram til informasjon og oppgaver. Mange foreslår bedre søkefunksjonalitet for å finne fram til rapporter, emner, osv. Bedre prosessstøtte gjennom mer funksjonalitet er også etterspurt, f.eks. mer/bedre søkefunksjonalitet, kunne lage egne rapporter og sende vedlegg i e-poster.
- **EpN - mer funksjonalitet og bedre brukervennlighet:** Forbedringer knyttet til funksjonalitet og brukervennlighet for emneredigering. EpN mangler funksjonalitet for at den skal tas i bruk i full skala. Den må takle emneredigering for flere semestre om gangen, ha (bedre) funksjonalitet for undervisnings- og vurderingskombinasjoner og tillatte korrekturlesning. EpN bør bli mer brukervennlig der arbeidsoppgaver kan utføres enklere og raskere, for eksempel gjennom standard oppsett/forhåndsdefinerte maler, forenkle samarbeid, innføre versjonskontroller.
- **Studentweb – bedre brukervennlighet:** De fleste forslagene går ut på bedre brukervennlighet og brukergrensesnitt. Studentweb oppleves lite selvforklarende og passiv. Studentweb bør bli mer aktiv mot studentene og oppmuntre til å utføre handlinger de må gjøre, spesielt i oppgaver med tidsfrister som oppmelding til undervisning, emner og betaling. Det må bli enklere å finne fram og forstå hvilke handlinger som skal utføres i tillegg til kontroller som opplyser om feil studenten gjør. Brukergrensesnittet (symboler, tekst, plassering av oppgaver) bør også være mer selvforklarende og tilpasses mobil bruk.
- **Fagpersonweb - mer funksjonalitet, integrasjoner og bedre brukervennlighet:** Faglærere trenger funksjonalitet for å kunne utføre oppgavene i Fagpersonweb. De trenger også mer data fra LMS systemer for å effektivisere oppgaver i Fagpersonweb. Det må bli enklere å finne fram til oppgaver og informasjon.
- **Søknadsweb – bedre brukervennlighet:** Bedre brukervennlighet knyttet til opptak. Det må bli enklere for studenten å finne det de ønsker å søke på. Datakontroller, veiledende tekster må hjelpe søkere i opplasting av dokumentasjon til søknader. Dokumentasjon fra gamle søknader må fjernes.
- **STAR – flere og mer intuitive rapporter:** Studieadministratorer har et stort behov for rapporter fra STAR, men de må bli mer brukervennlige. Det må bli lettere å finne rapporter, de må bli enklere å forstå, og det må være flere relevante rapporter
- **Vitnemålsportalen – videreutvikling av funksjonalitet:** Vitnemålsportalen bør utvikles for å inkludere fagskoler og visning av faktisk vitnemål, ikke innhold i vitnemål.
- **EVUweb – bedre brukervennlighet:** Det bør bli mulig for søker å foreta prioritering av søknader og navigering må forbedres.
- **Flyt – mer integrasjoner og funksjonalitet:** Flyt oppleves mangelfull mtp på integrasjoner og funksjonalitet. Den bør integreres til andre (arkiv)systemer enn P360 og utvides til behandling av andre oppgaver enn godkjenning, for eksempel klager og tilrettelegging.

Utdrag fra brukerundersøkelse for studenter

Detaljer i vedlagt Excel fil til sluttrapporten:
Unit - Vurdering av FS - Anonymisert - Brukerundersøkelse 032020 Studenter.xlsx

Studenter – respondentoversikt

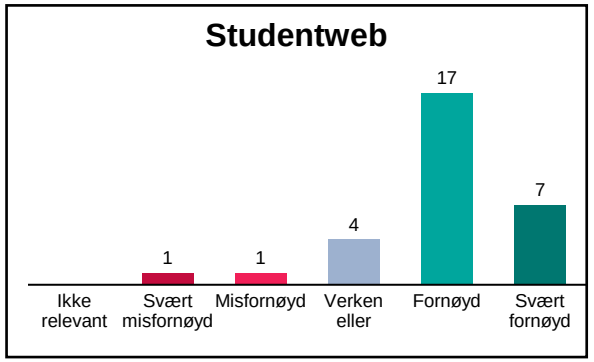
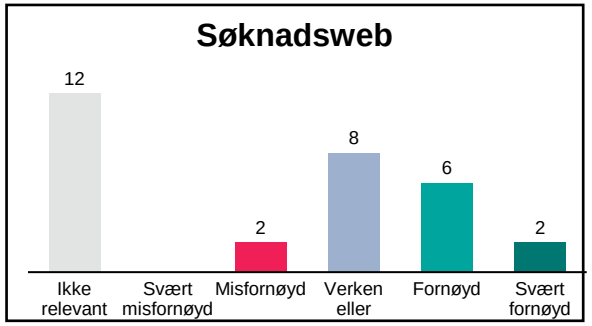
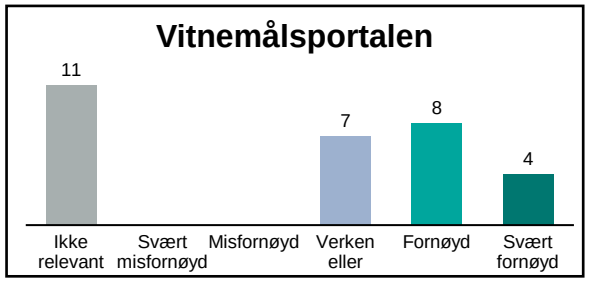
38 fra følgende institusjoner mottok brukerundersøkelsen

Institusjon
Høgskolen i Molde
Høgskolen i Østfold
Høgskulen i Volda
Høgskulen på Vestlandet
Høyskolen Kristiania
Kriminalomsorgens høgskole og utdanningssenter KRUS
MF vitenskapelig høyskole for teologi, religion og samfunn
Nord universitet
Norges teknisk-naturvitenskapelige universitet
OsloMet – Storbyuniversitetet
UiT Norges arktiske universitet
Universitetet i Agder
Universitetet i Bergen
Universitetet i Oslo
Universitetet i Stavanger
Universitetet i Sørøst-Norge
VID vitenskapelige høgskole

30 (79%) besvarte brukerundersøkelsen

Pågående studie	Antall år studert ved en høyere utdanningsinstitusjon i Norge			Delsum
	2 år	3 år	4-6 år	
Masterstudie		1	10	11
Bachelorstudie	7	4	7	18
Annet			1	1
Delsum	**7**	**5**	**18**	**30**

Studenter - oppsummering av tilfredshets FS applikasjoner

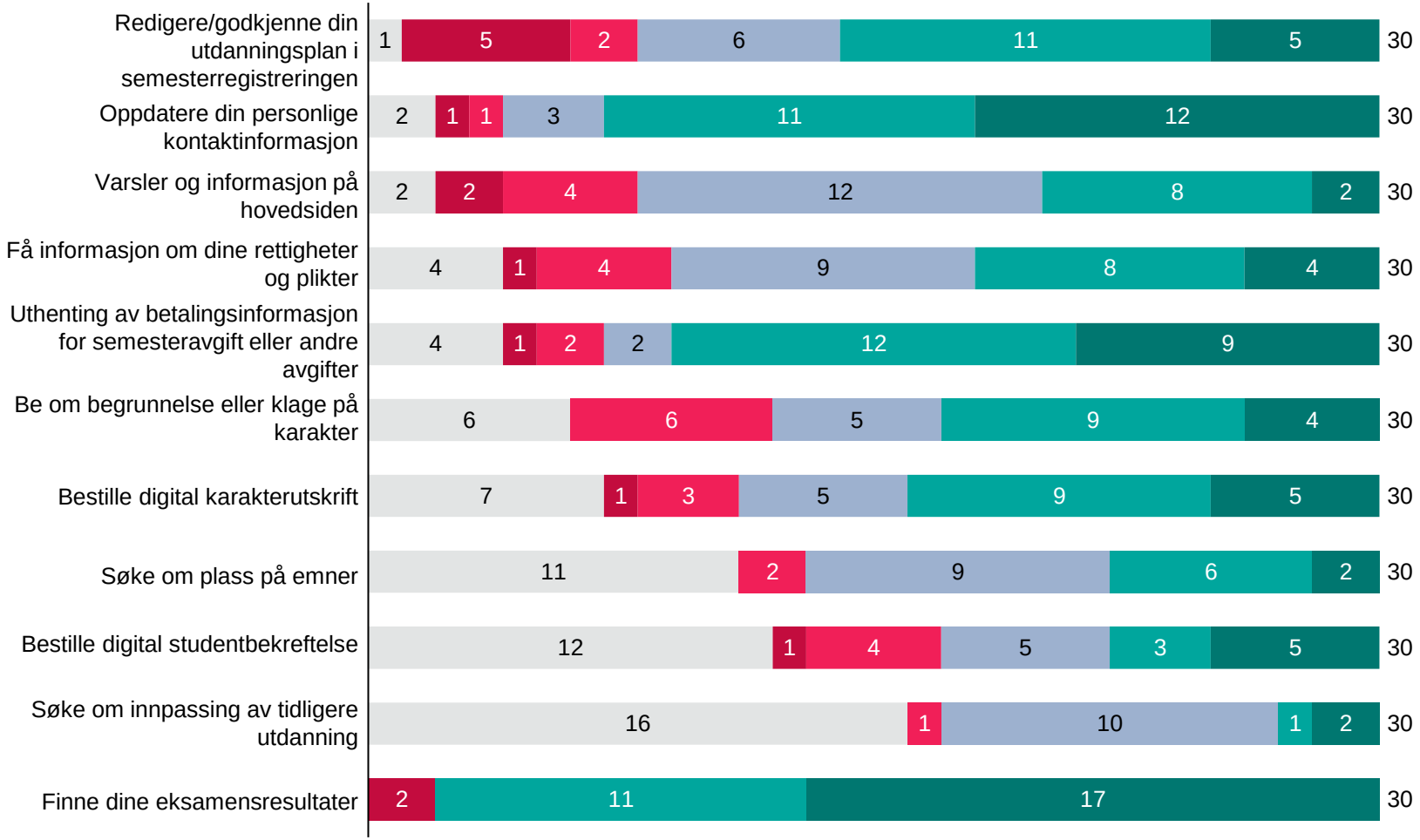
Hvor fornøyd er du med applikasjonen…	Kommentarer* fra (svært) misfornøyde…	Kommentarer* fra verken eller…	Kommentarer* fra (svært) fornøyde..
	• *Den virker noe tungvint å bruke enkelte ganger. Betalingsløsningene for studieavgift skulle vært mye mer brukervennlig, samt informasjon om klaging av karakterer og begrunnelse. Det er flere ganger jeg har måttet leite etter karakterer. Syntes også godkjenning av utdanningsplan er et herk, man går rundt og er redd for at man ikke har meldt seg opp til et fag.*	• *Syntes det var litt vanskelig å finne fram til studentstatus og karakterer da jeg ikke visste hva de ulike tingene der inne var for noe. Fikk aldri innføring eller gjennomgang av hva man brukte StudentWeb til.* • *Kunne ha vært tettere integrert med andre systemer* • *enkelte ting som man burde kunne gjøre selv, men der man må kontakte studieveileder*	• *Enkel og oversiktlig* • *Designet kunne vært utbedret for en lettere forståelse* • *Kan være utydelig om man har meldt seg opp riktig til fag.* • *Oppdatere kontaktinfo – var lett og brukervennlig* • *Bestille digital karakterutskrift* • *Ting kan ligge litt skjult og lagrer ikke foreløpige emneregisteringer*
	• *Alt for komplisert* • *Havner i loop ved innlogging fra Safari* • *Må ofte fylle ut adresse* • *Mange lurer på hvordan man legger til nye dokumenter i søknader* • *Feil søknadsstatus*	• *Helt greit, litt mye informasjon og tekst. Utseendemessig er det veldig gammelt og mye tekst.* • *Uoversiktlig, skjønte ikke helt hvordan bruke den på vgs.*	• *Kan være knotete for nye studenter* • *Kan være vanskelig å få god oversikt over emner og tilbud ved vertsuniversitet.* • *Vanskelig å finne allerede opplastet dokumentasjon.* • *Uklart hva som skal lastes opp av dokumentasjon i en søknad*
	Ingen kommentarer	Ingen kommentarer	• *Smart løsning for å slippe papirvitnemål* • *Rask respons ved forespørsler om vitnemål*

** Utdrag fra kommentarene i undersøkelsen*

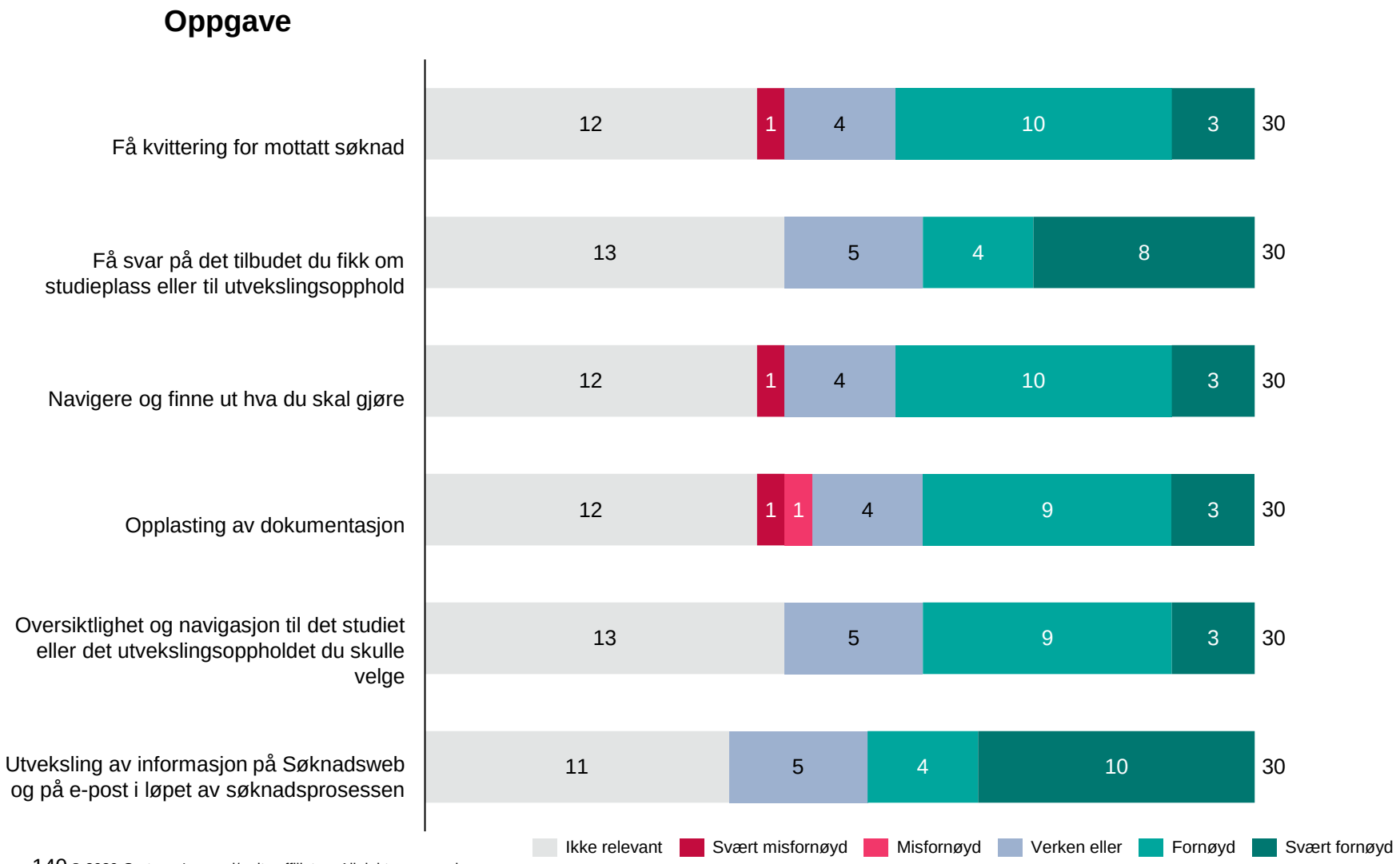
Gartner

Studenter - tilfredshet med oppgaver i Studentweb

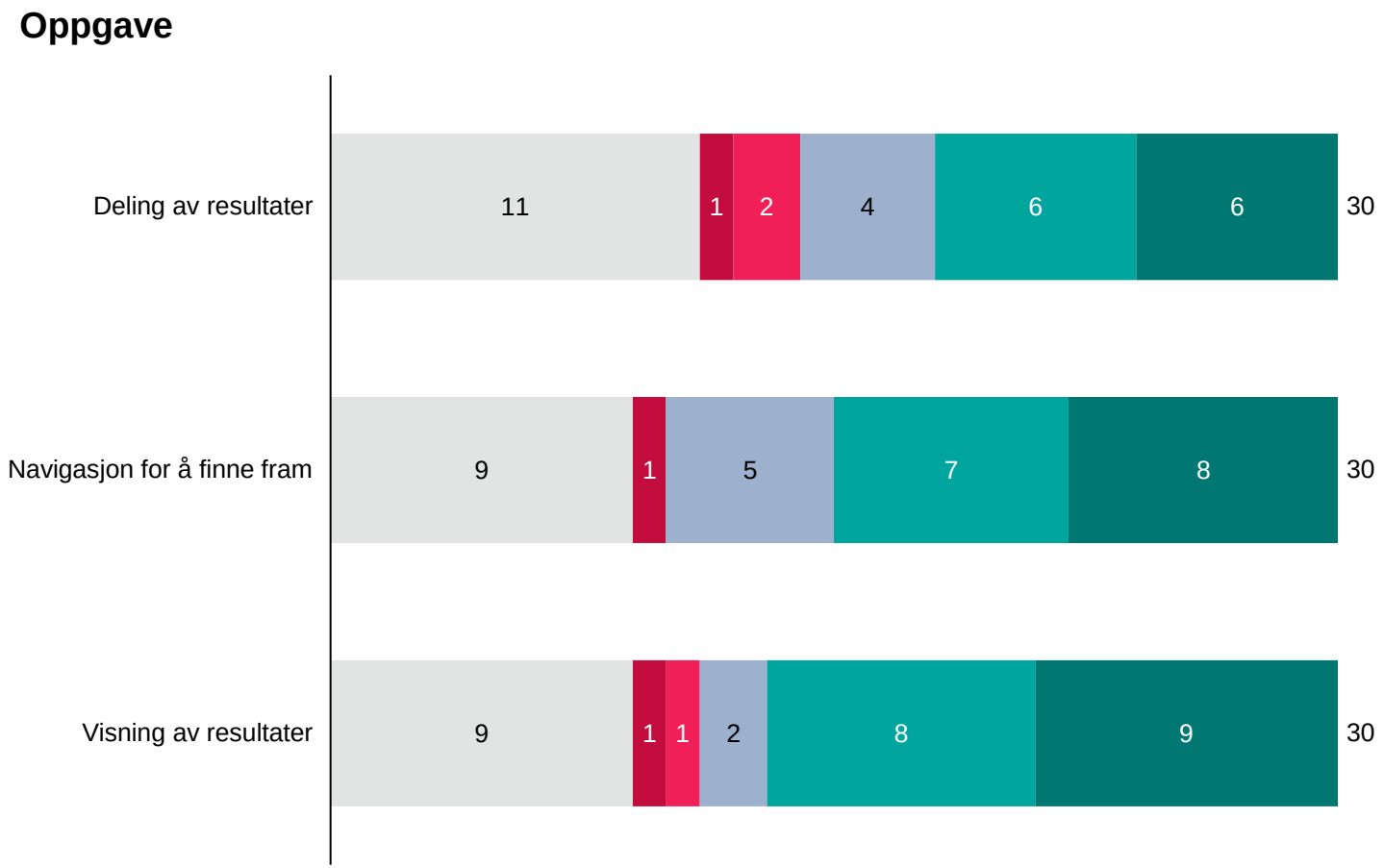
Oppgave



Studenter - tilfredshet med oppgaver i Søknadsweb



Studenter - tilfredshet med oppgaver i Vitnemålsportalen



Gartner

Studenter - forslag til forbedringer i Studentweb, Søknadsweb og Vitnemålsportalen

Nøkkelfunn om forbedringer i Studentweb, Vitnemålsportalen og Søknadsweb

- **Bedre brukervennlighet (navigasjon):** Det må bli enklere og raskere å navigere. Flere oppgir at oppgaver oppleves som gjemt og at sliter med å finne oppgaven eller må gjennom unødvendig mange klikk for å komme til en oppgave. De ønsker derfor å kunne nå oppgaver fra forsiden. Menyene oppleves også som lite intuitive og ønsker mer forståelige begrepsbruk og bedre struktur.
- **Mer moderne brukergrensesnitt**: Studentene ønsker mer moderne design – skrift, layout, symboler og farge. Det er nevnt at brukergrensesnittet ikke fungerer optimalt på mobil, og at dette er ønskelig å få til.
- **Mer datavalidering og veiledende tekster:** Mange feil studenter gjør i dag skyldes manglende informasjon og datakontroller av handlinger og opplastet dokumentasjon. Studentene ønsker mer veiledende tekster i systemet om handlinger som skal utføres, kontroll av dokumentasjon som skal lastes opp og kontroller av handlinger som må utføres. Det etterlyses også mer forståelige feilmeldinger.
- **Mer informative forsider og opplæring:** Flere studenter opplyser om at de ikke kjenner til flere av oppgavene som nevnes og ønsker mer informasjon raskt tilgjengelig og opplæring i mulighetene som finnes. Viktig informasjon og funksjonalitet bør være lettere tilgjengelig, for å sørge for at studentene får utført oppgavene sine i tide og raskere.

Spesifikke områder for forbedringer

- **Mer intuitiv redigering av utdanningsplan:** Nå er studenter usikker på om utdanningsplanen er riktig og ønsker en mer brukervennlig måte å administrere utdanningsplanen på. Det bør være mer veiledende tekster for å guide studenten, enklere å søke opp emner og det bør synliggjøres om studenten er meldt opp riktig, slik at de kan få en bekreftelse på at utdanningsplanen er rett.
- **Dokumentasjon i søknadsweb:** Fjerne dokumentasjon fra gamle søknader, veiledende tekst om påkrevd dokumentasjon slik at søker forstår hva slags dokumentasjon som skal lastes opp.
- **Begrunnelse og klage:** Integrere begrunnelse og klage i systemet slik at man unngår mail.

Utdrag fra brukerundersøkelse for fagpersoner

Detaljer i vedlagt Excel fil til sluttrapporten:
Unit - Vurdering av FS - Anonymisert - Brukerundersøkelse 032020 Fagperson.xlsx

Gartner

Fagpersoner – respondentoversikt

38 fra følgende institusjoner mottok brukerundersøkelsen

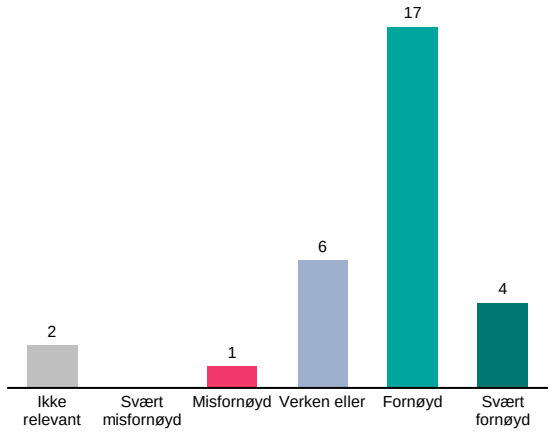
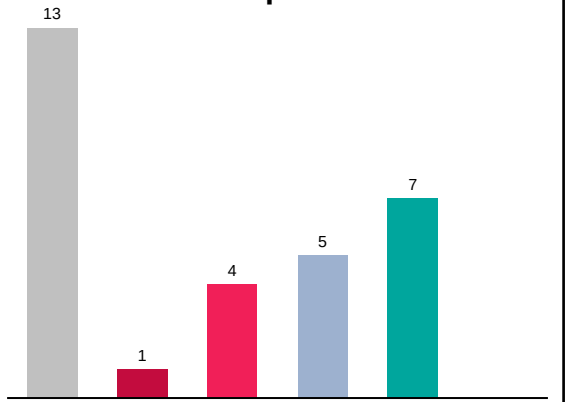
Institusjon
Høgskolen i Molde
Høgskulen i Volda
Høgskulen på Vestlandet
Høyskolen Kristiania
Kriminalomsorgens høgskole og utdanningssenter KRUS
MF vitenskapelig høyskole for teologi, religion og samfunn
Nord universitet
Norges teknisk-naturvitenskapelige universitet
UiT Norges arktiske universitet
Universitetet i Agder
Universitetet i Bergen
Universitetet i Oslo
Universitetet i Stavanger
Universitetet i Sørøst-Norge
VID vitenskapelige høgskole

30 (79%) besvarte brukerundersøkelsen

Antall års erfaring med bruk av Felles Studentsystem (FS) til studieadministrasjon	Antall respondenter
0 år	4
1 år	6
2 år	1
3 år	3
4-6 år	10
7-8 år	3
9+ år	3
Sum	**30**

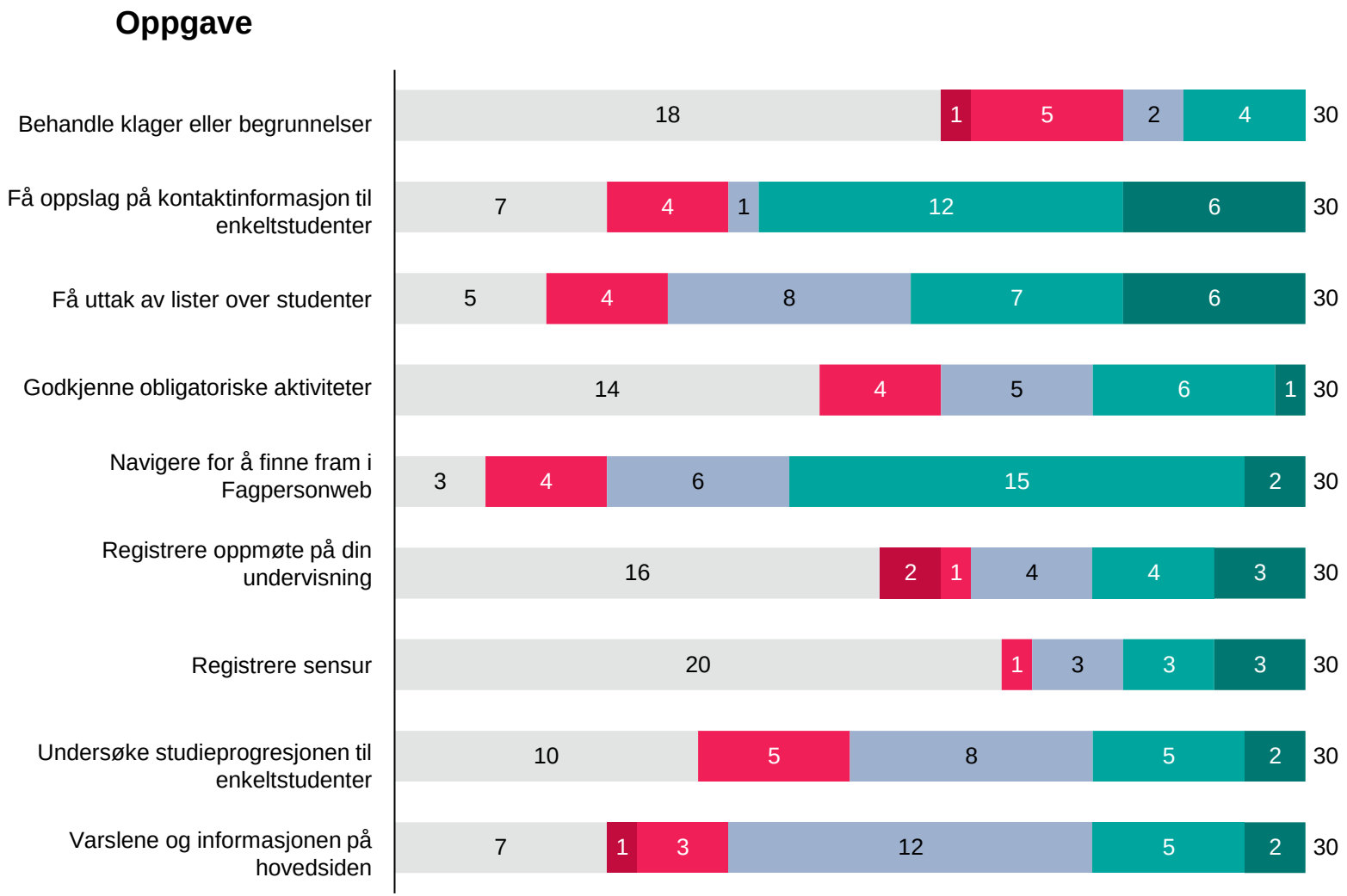
Gartner

Fagperson - oppsummering av tilfredshets FS applikasjoner

Hvor fornøyd er du med applikasjonen…	Kommentarer fra (svært) misfornøyde…	Kommentarer fra verken eller…	Kommentarer fra (svært) fornøyde..
Fagpersonweb 	• *Det er ikke mulig å differensiere oppmøte for studenter ut fra antall timer. Dette **gir en feilaktig oversikt over oppmøte.***	• *Ingen korrespondanse med Canvas. Mottar ikke meldinger om varsler per mail.* • *Noe **ulogisk brukergrensesnitt**, og det er lett å trykke feil når man skal inn og godkjenne oppgaver.* • *Der er ein del info som vi tidvis treng, som ikkje er der - t.d. adresse og karakterar* • *Innholdet er relativt ok, men jeg synes **brukergrensesnittet er tungvint**. Den glemmer også valg*	• *Den gjør registering av store kull (mer enn 100 studenter) mer effektiv og gjennomførbar.* • *Skulle veldig gjerne sett at **systemet snakker mer med andre systemer**. Gjør en del Dobbeltføring, noe som føles unødvendig.* • *Enkelt system å bruke. Synes det er litt skjult med meny helt oppe i feltet, **for nye personer blir det ofte spørsmål om layout**.* • *Lett å holde oversikt over emnene en har.* • *Gjerne **få større/uthevet skrift** på de forskjellige oppgavene man registrerer (Oblig. arbeidskrav, ol.)* • *En kan også oppleve at **studentlista på Fagpersonweb og Canvas ikke stemmer***
EpN 	• *Tungt system med veldig mye klikking og åpning/lukking av faner. Intuitivt og greit i utgangspunktet, men dette kan ha noe med hvordan institusjonen har satt det opp. Jeg er dessuten veldig kritisk til at det tilsynelatende ikke er versjonskontroll og at veldig mange kan gå inn og endre uten at det sees åpent.* • *Det største problemet med EpN er at **du ikke kan stole på at teksten er identisk med de endringene du innførte året før**. Har opplevd å legge inn endringer ett år, og neste år er disse endringene fjernet igjen. Dette er utrolig frustrerende.* • ***Tungvint, lite selvinstruerende, begrepsbruk lite samsvarende med andre begrep som brukes i institusjonen.***	• *Ikke så intuitivt **brukergrensesnitt*** • *Altfor mange bokser som man må inn i. Hadde vært bedre å redigere en et program som så ut som emnebeskrivelsen på nett.* • *Savner **funksjonaliteter for å lettere få opp lister**, og kunne tenke meg et malverk som produserer ferdige emnebeskrivelser organisasjonen kan designe selv i rapport-funksjonen.*	• *Fungerer greit. En ting som kan utbedres er at det **kommer opp en mer detaljert feilmelding** når det er noe som ikke er utfylt eller er fylt ut feil, slik at man kan få veiledning av systemet på hvordan det fylles ut riktig* • *Brukergrensesnittet er OK, men jeg savner muligheten for å **lagre lister med relevante emner**. Det er en mulighet for å lagre lister, men denne blir ikke bevart.* • *Enkelt å gjøre endringer i emneinfo. Men det kunne godt **være litt mer veiledende informasjon** i EpN - eller lenker til slik informasjon.*

Gartner

Fagpersoner - tilfredshet med oppgaver i Fagpersonweb



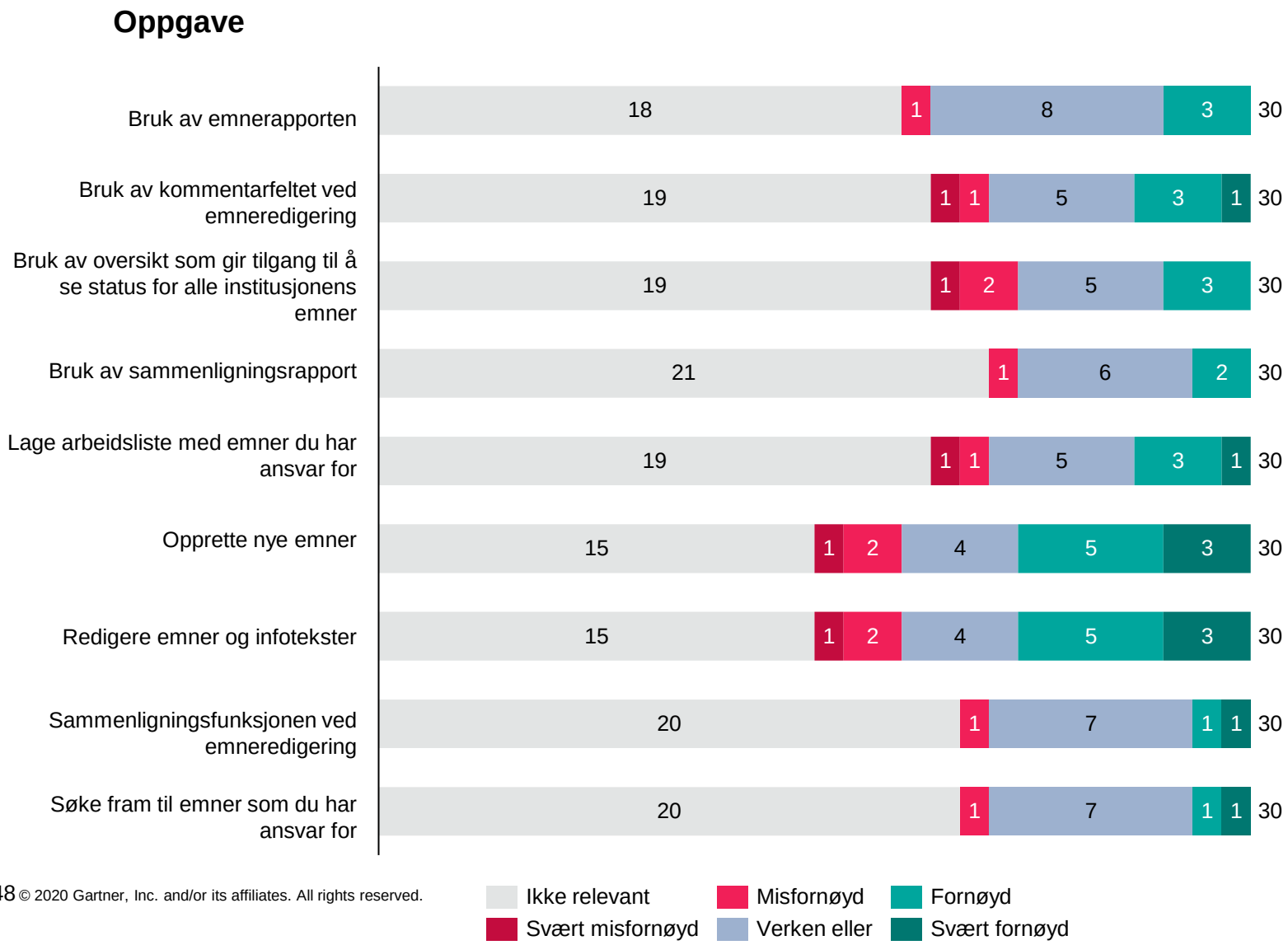
Fagperson - forslag til forbedringer i Fagpersonweb

Har du forslag til hvordan Fagpersonweb kan forbedres for å forenkle dine oppgaver knyttet til administrasjonen av undervisning og vurdering?

- **Behov for data/informasjon gjennom integrasjoner:** Delvis overlappende funksjonalitet og informasjon i systemer gjør at faglærere får utfordringer med å holde oversikt og må dobbeltføre. Integrasjoner for oppgaver og informasjon som kan utføres/finnes i flere systemer er ønsket som forbedring. Bedre kommunikasjon med LMS plattformer framheves som et stort behov (Canvas spesielt).
- **Behov for data/informasjon:** De fleste behovene knyttes til behov for funksjonalitet som gjør at det kan tas ut mer data/informasjon i systemet, f.eks. kunne ta ut epostliste med private e-postadresser.
- **Behov for funksjonalitet:** Det oppgis at Fagpersonweb er mangelfull når det kommer til funksjonalitet uten at det uttrykkes nærmere hva det gjelder, og at den derfor ikke brukes.
- **Brukervennlighet generelt**: Faglærere oppgir frustrasjon lav brukervennlighet totalt sett for studieadministrative oppgaver innenfor undervisning og vurdering. Det er for mange systemer å forholde seg til og ønsker seg færre systemer å forholde seg til innenfor studieadministrasjon.
- **Brukergrensesnitt**: Det må bli enklere og raskere å finne fram til informasjon og funksjonalitet. Bedre menyvalg med mer logiske navn. Navigasjonen må gå raskere med færre klikk. Utheving av tekst når man klikker på noe. Mindre scrolling opp og ned.

- **Oppmøteregistering:** Flest ønsker forbedringer innenfor oppmøteregistering. Fagpersonweb må snakke bedre med andre systemer som registrerer oppmøte for å hindre dobbeltarbeid. Det er også behov for mer granuler funksjonalitet for korrekt oppmøteregistering, samt ønske om mer effektivisering av godkjent oppmøte. Det er også ønske om enklere oppsett fra fagpersoner, siden dagens prosess med at studieadministratorer setter det opp oppleves tungvint.
- **Godkjenne obligatoriske aktiviteter:** Overføring av informasjon om godkjenning som finnes i andre systemer (f.eks. Canvas) til Fagpersonweb må bli bedre for å redusere unødvendig tidkrevende og manuelt ekstraarbeid. Det er også ønske om mer funksjonalitet; f.eks. registrere godkjenninger på studenter som har meldt seg av. Automatisering av godkjenninger mtp. oppmøtekrav og godkjente innleveringer bør gjøres av systemet.
- **Undersøke studieprogresjon:** Det er ønsker om å gi emneansvarlige (enklere) tilgang til studieprogresjon, studenters utdanningsplan og informasjon om studenters oppmeldinger. Dette innebærer også færre klikk og mer synlig informasjon.
- **Klager, begrunnelser og sensur:** Faglærere har problemer med at oppgaven kan utføres i andre systemer som MittUiB og Inspera. Det er derfor ønske om å kunne fortsette å bruke Inspera eller MittUiB.

Fagperson - fordeling av tilfredshet med oppgaver i EpN



Gartner

Fagperson - forslag til forbedringer i EpN

Har du forslag til hvordan EpN kan forbedres for å forenkle administrasjon av emner?

- **EpN i sin helhet:** Flere etterspør bedre brukervennlighet og brukergrensesnitt. Dette går ut på bedre struktur i EpN, mer bruk av veiledningstekster og mer forståelige feilmeldinger.
- **Redigere emner:** Mulighet for å lime inn andre format, som punkttekster. Ulike versjoner av emnebeskrivelser oppleves som et problem, og versjonskontroll er derfor foreslått.
- **Sammenligningsrapport**: Burde vært mulig å genere en slik på alle stadium i prosessen
- **Lage arbeidsliste med emner du har ansvar for:** Listen bør generes automatisk for de som skal redigere emner. lagres for senere bruk. Nå ser emnelisten for en fagperson til å være urelevant og tilfeldig plassert og det er vanskelig å forstå hva som er relevant for en fagperson.
- **Oversikt**: Bør være mulig å dele inn oversikten etter program for å gjøre oversikten lettere for personer som har mange emner.

Vedlegg

IT-direktører fra følgende 5 institusjoner deltok i intervju ifb. med funksjonell vurdering

	Universitet	
1	NTNU (BOTT)	✓
2	Universitetet i Bergen (BOTT)	✓
3	Universitetet i Oslo (BOTT)	✓
4	Universitetet i Tromsø (BOTT)	✓
5	OsloMet – storbyuniversitetet	✓

Identifiserte relevante målbilder fra Digitaliseringsstrategien og Handlingsplanen for innspill til utarbeidelse av fremtidig målarkitektur

Digitaliseringsstrategi for universitets- og høyskolesektoren 2017-2021

3.2.5 MÅLBILDE FOR DATA OG INFRASTRUKTUR (S.11)

- Data lagres én gang og tilgjengeliggjøres fra én kilde.
  - *Fokus på konsistens*
- Data er gjenfinnbare, tilgjengelige, interoperable og gjenbrukbare, iht. FAIRprinsippet.
- Infrastrukturen er fleksibel og legger til rette for mobilitet og utvikling.
  - *Infrastrukturen defineres som FS plattform i applikasjonsutvikling*
  - *Tilrettelegger for utvikling av eksterne både i og utenfor sektoren*
- Helhetlig styring og ledelse av informasjonssikkerhet er et fundament for digitalisering og strategiske satsinger, og bygger opp under sektorens målsetninger.
  - *Henvises til aktuelle målbilder i handlingsplanen*

Fra handlingsplan for digitalisering i høyere utdanning og forskning 2019-2021

7 INFORMASJONSSIKKERHET OG PERSONVERN (s.25)

7.1 MÅLBILDER

MIS1 Helhetlig styring av informasjonssikkerhet og personvern er et fundament for digitalisering og strategiske satsinger.

MIS2 Institusjonene har egeninteresse i å løfte kravene til informasjonssikkerhet og personvern høyere enn de nasjonale minstekravene, basert på informasjonens verdi.

MIS3 Ledelsen ivaretar institusjonenes informasjonsverdier, og følger nasjonale føringer gjennom et systematisk arbeid for tilfredsstillende informasjonssikkerhet og personvern.

MIS4 Forskere ivaretar informasjonssikkerheten til data, herunder forskningsdata, og forskningsdeltakernes personvern på en tilfredsstillende måte.

MIS5 Studenter og ansatte er i stand til å håndtere data slik at hensynet til informasjonssikkerhet og personvern ivaretas.

MIS6 All bruk av data og IKT ivaretar hensynet til informasjonssikkerhet og personvern gjennom hele livsløpet (innebygd informasjonssikkerhet og personvern).

Tekst i kursiv er fra diskusjonsmøte

Gartner

Definisjoner/forklaringer på Gartners topp 10 trender for høyere utdanning

Top 10 Business Trends	Definition/Description
Analytics Everywhere	In higher education Analytics refers to the collection and analysis of data to provide insights that can guide actions that improve outcomes and efficiency.
New Business Models	Higher education is introducing new business models in response to disruptive forces threatening long-established business models and the digital transformation of society.
Reinventing Credentials	Offering credentials that are stackable, portable, transparent and durable. This includes but is not limited to, degrees, certificated, industry certifications, licenses, badges, apprenticeships and micro credentials.
Online Differentiation	The ability of an institution to demonstrate a compelling, distinctive online learning offering.
Ethical Use of Data	In higher education this comprises the systems of values and fair principles for the conduct of electronic interactions between people, things and the institution. Includes social and mobile technologies, social interaction, cloud and security, big data an privacy, AI, and predictive algorithms.
Corporate Collaboration	Engaging with corporate sector to broaden how higher education attracts new students to help retain and upskill current workers. Also includes being more proactive and receptive to many new forms of corporate partnerships, including co-branding initiatives.
Ecosystem	Changes that are catalyzed by the breaking down of traditional boundaries surround the academy. The result is a more fluid business environment that extends the value chain to the larger education ecosystem.
Collegiate Esports	Involves playing computer games against others on the internet, often for prize money. These competitions are often watched by other people using the internet or sometimes as special organized events. Recently, it has emerged as a program at the collegiate level, moving beyond an established club or recreational activity.
Digital Dexterity	The ambition and ability of employees to apply technology for better business outcomes. In higher education, it includes creating and interacting with new forms of media and technology to create, discover, interpret and disseminate information,
Creative Financing	The use of financial instruments to fund higher education. These include arrangements such as income share agreements and employer sponsorships.

Source; *Top 10 Business Trends Impacting Higher Education in 2020* (Gartner Research ID: G00465205)

Gartner

Definisjoner/forklaringer på Gartners topp 10 strategisk teknologi for høyere utdanning

Top 10 Strategic Tech	Definition/Description
Artificial Intelligence Strategy and Tactics	A discipline that applies advanced analysis and logic-based techniques, including machine learning, to interpret events, support and automate decisions, and take action. AI is a general-purpose technology.
Next-Generation Security and Risk Management	Next-generation security and risk management (SRM) in higher education focuses increasingly on end-user trust and, thereby, goes beyond the fundamentals of resilience and central control.
Smart Campus	A smart campus is a physical or digital environment in which humans and technology-enabled systems interact in increasingly open, connected, coordinated and intelligent ecosystems.
Nudge Tech	Collection of technologies that work together to achieve timely, personalized interaction with students, staff and faculty, such as a just-in-time text (SMS) reminder for class. Technologies used include chatbot, texting, algorithmic analytics, machine learning and AI CI.
Digital Credentialing Technologies	Digital credentials enable institutions to deliver authenticated credentials instantly and enable learners to control and share credentials more freely in the education ecosystem.
Cross-Life-Cycle CRM	CRM deployment that creates an enterprisewide, 360-degree view of a constituent, most often a student, across the major life cycle phases, beginning with precollege and moving through prospect, applicant, enrolled, graduated and alumni statuses.
5G/Ecosystem Infrastructure	5G service comprises local or wide-area cellular data connectivity based on the next generation of mobile and fixed wireless communications networking.
New Display, Visualization and Collaboration Technologies	With the increased adoption of online learning, new styles of display, collaboration and visualization technologies are combining in higher education to create new forms of engagement in online classes and online breakout rooms.
Career Software	Colleges and universities use career software to assist students in planning for their post-tertiary education careers and life workforce, including assessing their skills and making connections with employers.
Faculty Information Systems	An FIS tracks all aspects of faculty data so that the institution can maintain a single source of truth for faculty members on their credentials, careers, teaching, research and support aspects of faculty personnel administration.

Source; *Top 10 Business Trends Impacting Higher Education in 2020* (Gartner Research ID: G00465205)

Gartner

Ordliste

Ord	Definisjon
Superbruker/FS bruker	Studieadministrativt ansatt. Ansvarlig for å administrere studentlivsyklusen (opptak, kvalifikasjon osv.). Er sluttbruker av FS og representerer FS-bruker eller FS superbruker i den funksjonelle vurderingen av FS.
Student	Nåværende student ved en utdanningsinstitusjon i Norge som bruker en av applikasjonene i FS systemet, eksempelvis er sluttbruker av Studentweb.
Fagperson	Fagpersoner i FS opptrer i ulike roller som f.eks. lærere eller ansvarlige for undervisning, eller som sensorer i forbindelse med eksamen, eksempelvis er sluttbruker av Fagpersonweb.

Kontaktpersoner

Gartner
Erik Lehne
Senior Managing Partner
Public Sector Norge
Gartner Consulting
Telefon:+47 906 53 781
erik.lehne@gartner.com

Gartner
Baber Nisar
Director
Public Sector Norge
Gartner Consulting
Telefon:+47 938 70 637
baber.nisar@gartner.com

