Medborgarnavets plug-in-arkitektur

Innehåll

Versionshistorik:

2020-12-28	Sven-Håkan Olsson	Dokumentet etablerades.
2021-02-01	Sven-Håkan Olsson	Mindre tillägg.
2021-06-07	Sven-Håkan Olsson	Utvidgning för plug-ins för infokällor, mm.
2021-06-07	Sven-Håkan Olsson	Lite mer om visning, mm.
2021-06-07	Sven-Håkan Olsson	Småredigeringar, lite mer om ansvar och säkerhet.
2021-08-16	Sven-Håkan Olsson	Mindre ändringar.
2021-11-26	Sven-Håkan Olsson	Mindre ändringar.
2021-11-28	Sven-Håkan Olsson	Mindre ändringar.
2021-11-30	Sven-Håkan Olsson	Mindre ändringar. Utgör utgåva A.

Bilaga 1_1 Medborgarnavet_PlugInArk_utgåva_211130.docx

1. Inledning och bakgrund

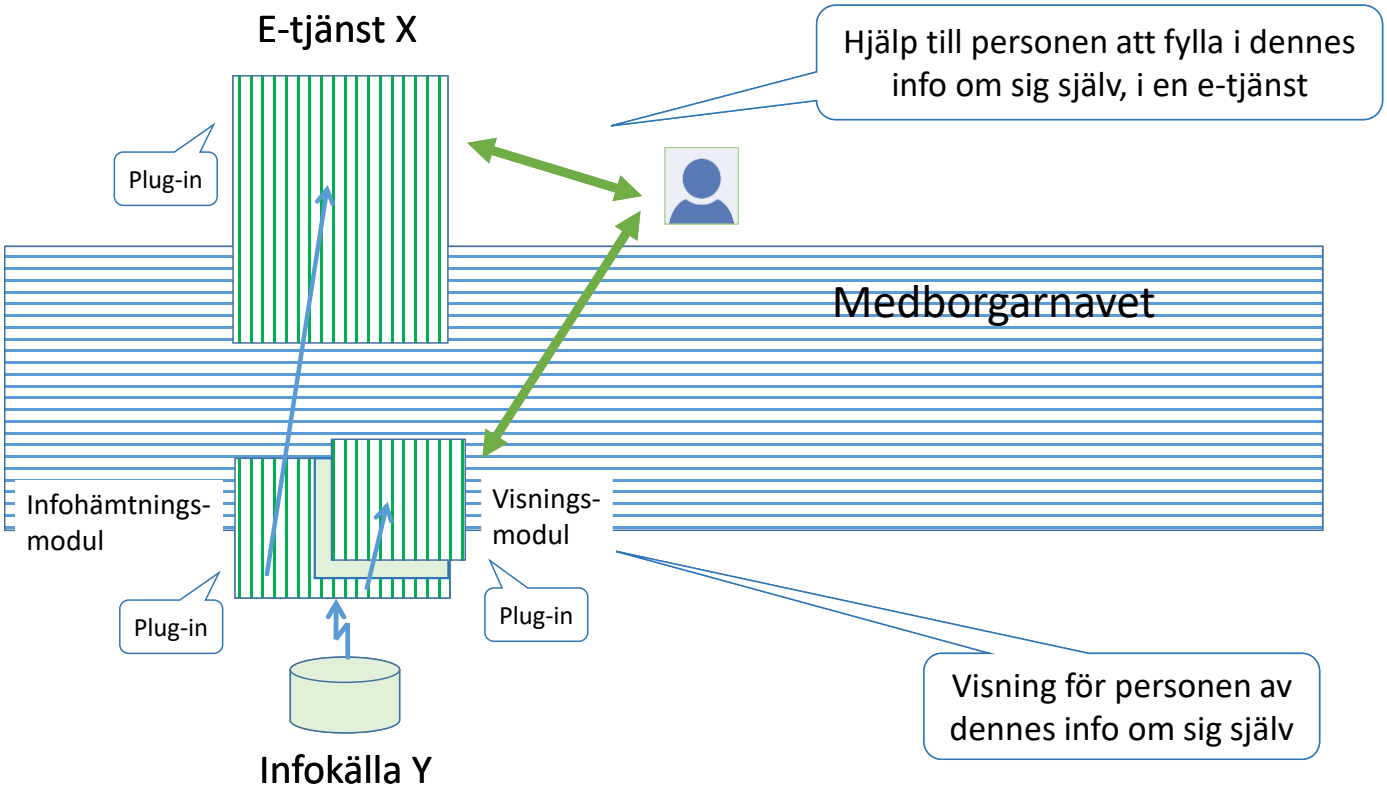
Medborgarnavet är tänkt att ge medborgaren tillgång till information om sig själv ifrån offentlig sektors register, samt ifrån andra register såsom inom bankvärlden.

Redan ändamålet att kunna se sin egen information är viktig ur ett transparens- och demokratiperspektiv samt för att kunna påpeka eventuellt ofullständig information.
Men projektet syftar också till att medborgaren lätt ska kunna använda lämpliga delar av informationen för att fylla i diverse e-tjänster (e-blanketter, e-ansökningar etc) som kan komma ifråga i olika sammanhang. Egen central registerinfo ska vara ett verktyg i medborgarens hand.

Hur ska då olika informationselement från registren kunna landa i dessa e-tjänster eller visningsmoduler? Det finns tyvärr ingen allmänt använd standard eller metod för detta. Projektet vill därför ta ett initiativ att utforska sätt att skapa samverkan mellan e-tjänster och en informationstjänst såsom Medborgarnavet. Vi tänker oss en s.k. plug-in-arkitektur – att exempelvis en e-tjänst ska kunna vara en plug-in som passar in i informationstjänsten så att datafält på ett ordnat och säkert sätt ska kunna överföras.

Nedan visas de två huvudsakliga användningsaspekterna i **ett** enkelt exempel:

- **Visning** av info om en själv
- Hjälp att **fylla i e-tjänster** med hjälp av data om en själv

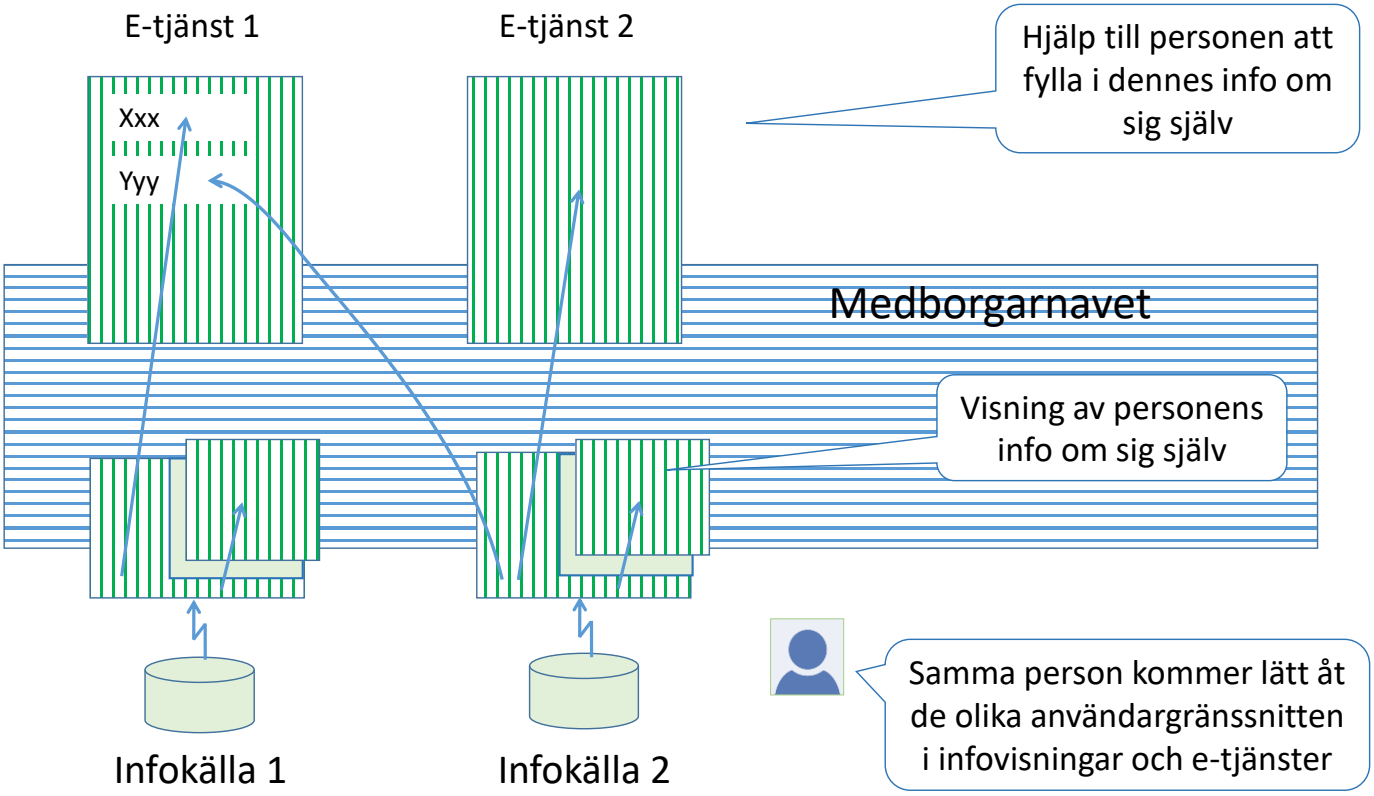


Medborgarnavet utvecklas i sin första inkarnation som ett projekt inom Vinnovas CivicTech-program 2020-2021. Ett av exemplen på användningsfall där är för e-tjänsten Valter tillsammans med hämtning av bankdata (detta handlar om God mans årsredovisning till överförmyndaren). Ett annat är eAnsökan EkBist samt Multifråga, tillsammans med infotjänsten SSBTEK (vilket handlar om att söka ekonomiskt bistånd)

2. Övergripande arkitektur

Huvudidén med Medborgarnavet är alltså att ge medborgaren själv tillgång till sin information, inte bara för att kunna **se den**, utan också för att enkelt kunna **använda** informationen. Därför behövs ett koncept för flexibel och teknikoberoende integration och datakommunikation mellan e-tjänster, informationskällor, visningsmoduler och Medborgarnanavet i sig. Vi ser alltså detta som en plug-in-arkitektur.

Arkitekturen ska förhoppningsvis kunna vara framtidssäker genom att olika nya delar ska kunna tillföras.



Vi räknar därför med tre sorters plug-ins:

- **Infohämtningsmoduler gentemot infokällor**
- **Visningsmoduler**
- **E-tjänster**

Medborgarnavet i sig är den gemensamma plattformen som samordnar och integrerar olika sorters plug-ins.

Viktigt att framhålla är att inga nya **offentliga** register normalt skapas – data lagras endast temporärt, under visning/överföring. S.k. Eget utrymme ska dock kunna förekomma, där personen under eget ansvar exempelvis kan lagra en halvifylld e-blankett och fortsätta fylla i imorgon, stuva om information för visning osv.

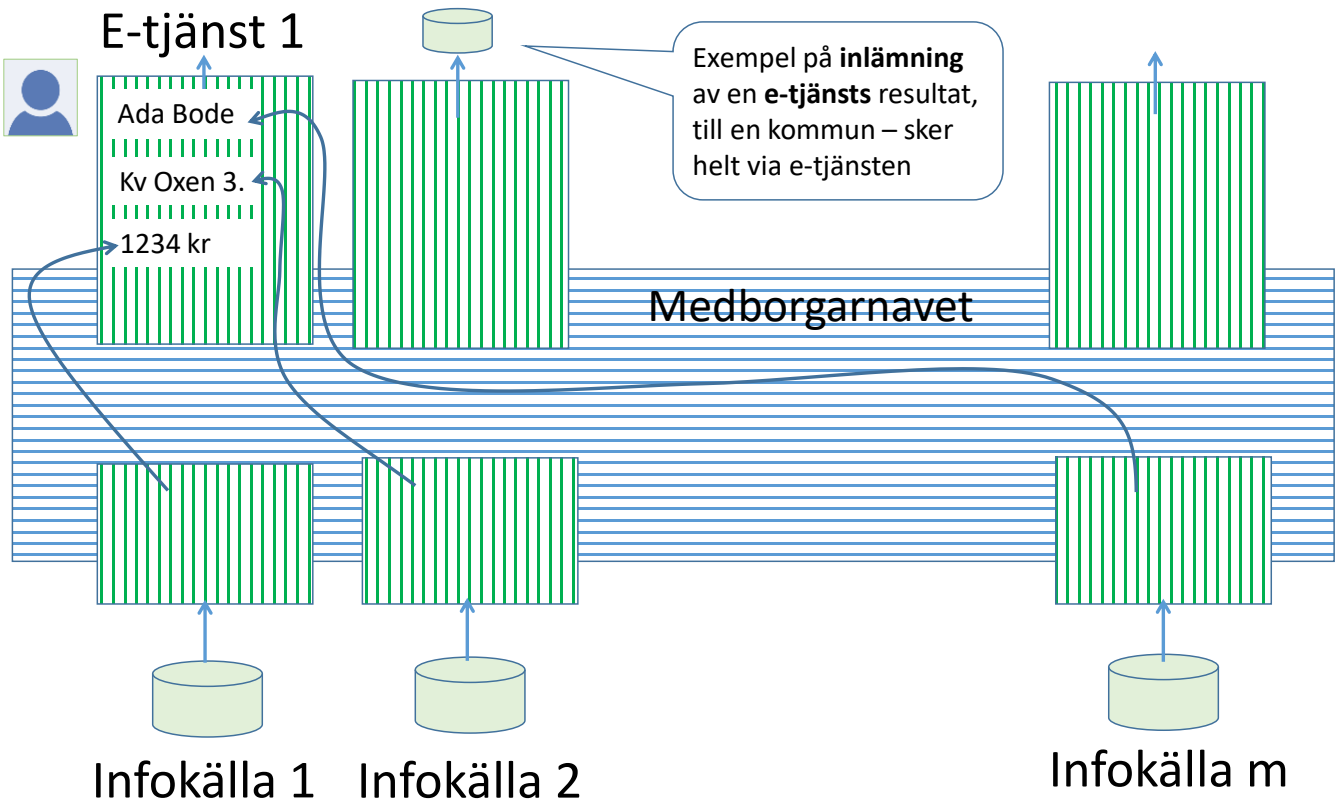
Medborgarnavsprojektet förespråkar för övrigt ett mer övergripande begrepp, Privat skyddat utrymme, för att slippa vissa juridiska komplikationer med begreppet Eget utrymme (se huvudrapporten från Medborgarnavsprojektet).

Man skulle kunna tänka sig att ett kommersiellt integrationsnav (eller open source) skulle kunna användas som ett Medborgarnav, såsom BizTalk, MuleSoft, olika s.k. ESB-sviter mm. Dock bedömer vi att fördelarna med ett färdigt integrationsnav i detta fall inte överskrider nackdelarna (hög inlärningströskel, arkitekturen blir mer komplex, licens- och/eller supportkostnader, att adaptering till e-tjänster/infokällor/visningsmoduler ändå måste göras och att detta är ofta den dyra delen, mm).

(I vissa framtida sammanhang kan man även tänka sig att det inte endast handlar om infokällor, det skulle även kunna röra sig om den andra riktningen, infomottagare.)

2.1. Förifyllnad i e-tjänst

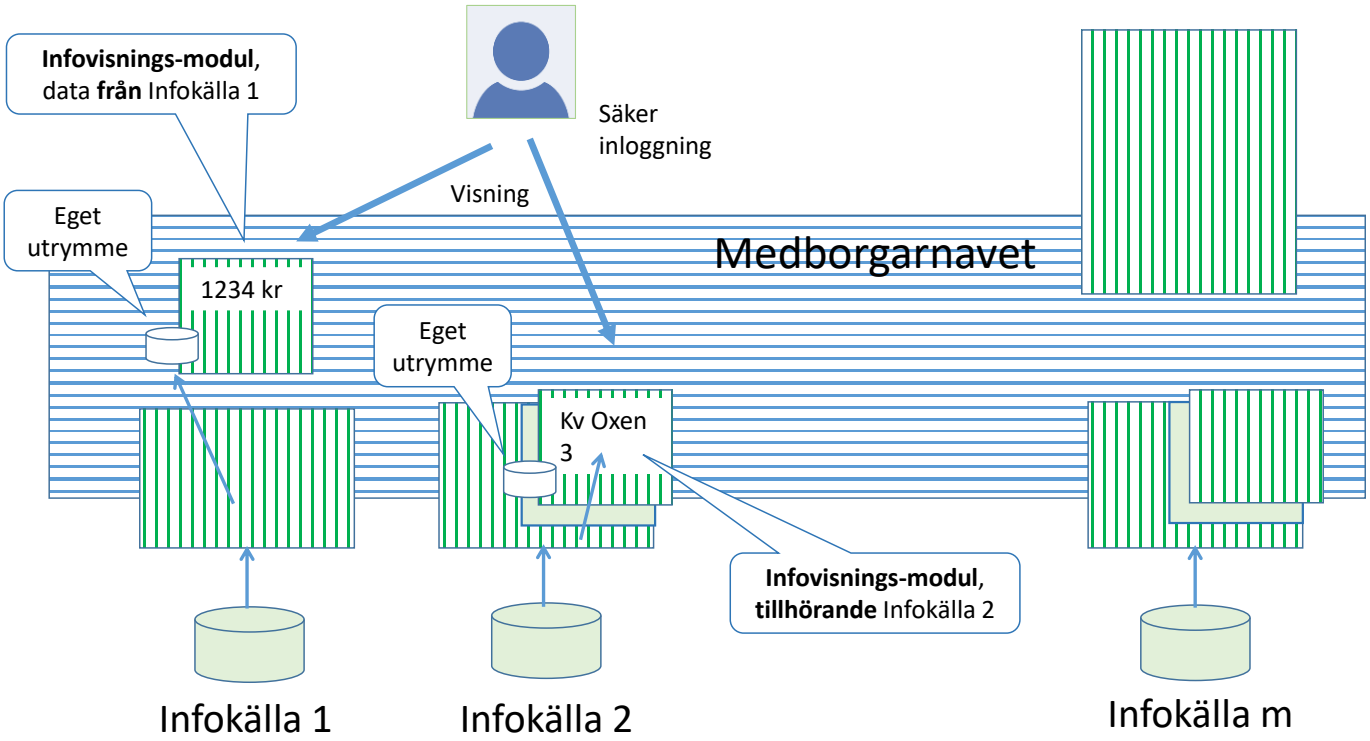
På nedanstående sätt ska automatisk *förifyllnad i olika e-tjänstefält* (i ett e-formulär) kunna ske, baserat på information som redan existerar om medborgaren i olika centrala register (därefter kan inlämning av e-formuläret ske av användaren till kommunen):



I andra fall kan det snarare än direkt automatisk förifyllnad handla om att erbjuda en palett av informationsfält som användaren successivt väljer ut till specifika blankettfält, eller om andra mer eller mindre avancerade sammanställningsmöjligheter som kan utföras inom Eget utrymme.

2.2. Visning av egen info

Förutom att betjäna e-tjänster där i viss mån personens egen info kan visas, är förstås en av huvudprinciperna hos Medborgarnavet att erbjuda *visning av central information om personen för personen själv*, varför visningsmoduler även behövs i arkitekturen:

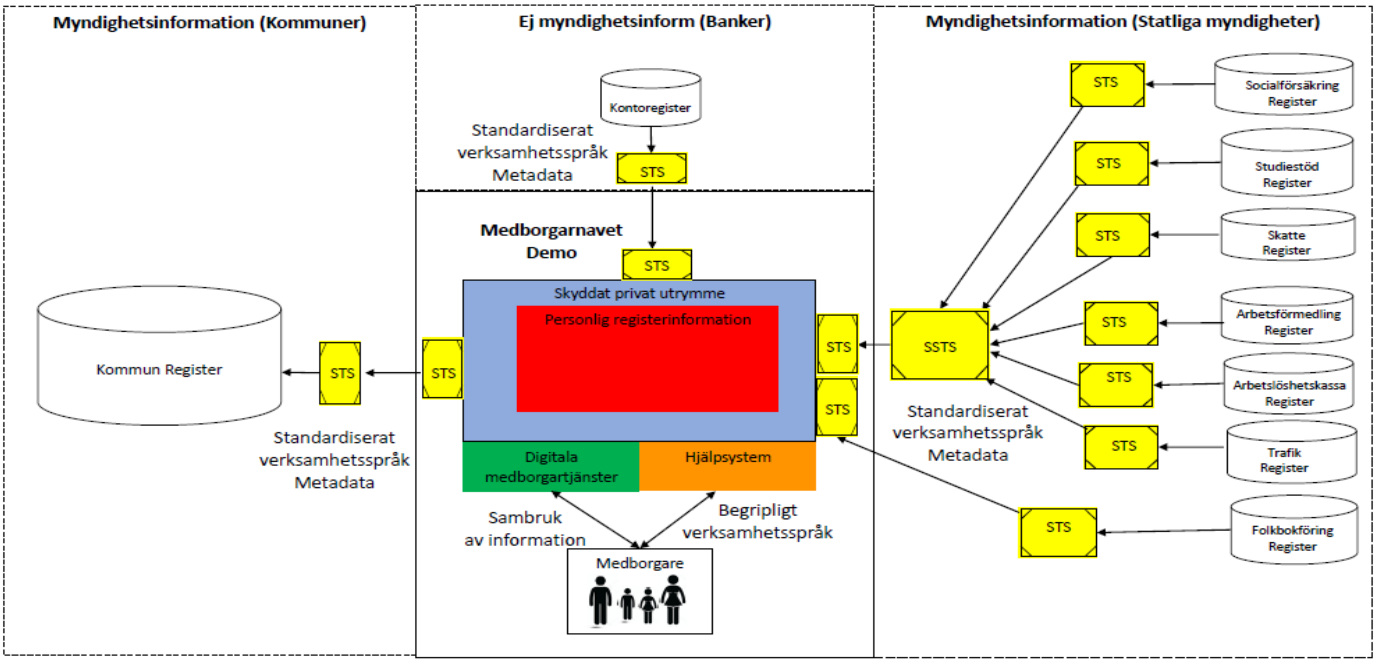


Denna infovisning ligger ibland inom "black box-ansvaret" för en viss infokälle-plug-in, varför en sådan plug-in kan sönderfalla i två delar, en för visning, en för informationshämtning. I andra fall finns infovisningsmodulen separat inom Medborgarnavet.

Efter säker inloggning i Medborgarnavet ska användaren således kunna välja olika visningsmoduler med data om sig själv.

2.3. Koppling till överliggande koncept

I projektets Vinnova-ansökan inkluderades följande principbild som sammanfattar ovanstående avsnitt:

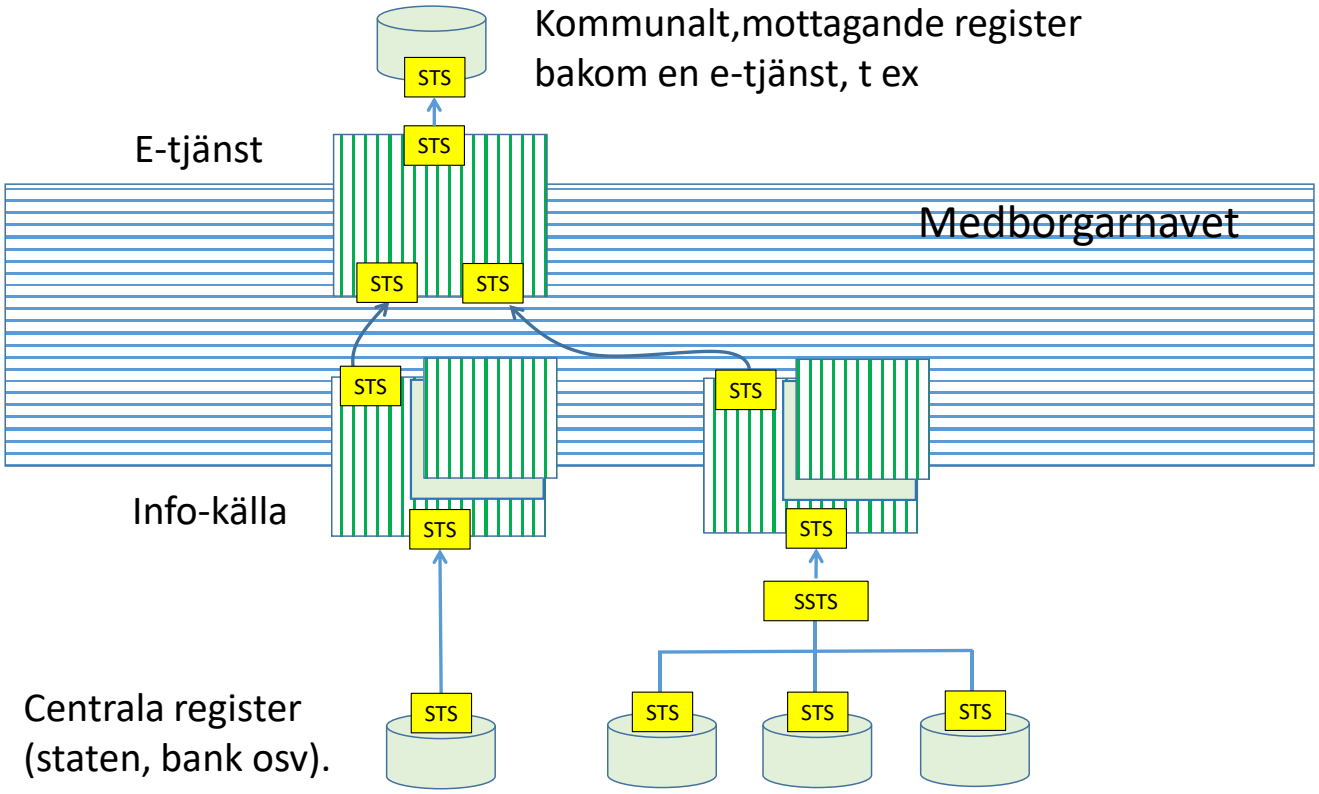


Plug-in-scenariona i föreliggande skrift är således en detaljering av delar av denna principbild.

Framförallt är detaljeringen en inzoomning på den gröna rutan "Digitala medborgartjänster", tillsammans med den röda rutan "Personlig registerinformation" (samt i viss mån även den blå rutan "Skyddat privat utrymme").

I plug-in-scenariona förenklas det dock så att "Centrala register" omfattar principbildens "Myndighetsinformation (Statliga myndigheter) respektive "Ej myndighetsinformation (Banker)".

Plug-in-scenarionas pil "Inlämning" påvisar kopplingen till principbildens "Myndighetsinformation (Kommuner)". Som detaljering markeras nedan även de "gula" byggstenarna STS (System-till-system-API[1]) och SSTS (Sammansatt-system-till-system-API) i plug-in-arkitekturen:



[1] API – Application Programming Interface – maskingränssnitt.

2.4. Hur tät koppling via plug-in-tekniken?

En mycket viktig grundprincip har varit att skapa så "lös koppling" som möjligt mellan olika plug-ins (samt med Medborgarnavets inre). Med det menas att man inte ska behöva göra antaganden om att en plug-in är utvecklad i samma programmeringsmiljö som en annan, inte heller att de körs i samma exekveringsmiljö eller i samma server, etc. Utan lös koppling skulle flexibiliteten minskas i en arkitektur som denna.

Varje plug-in ska kunna ses som en "black box" där de som har förvaltningsansvaret för boxen ansvarar för innanmätet, medan andra plug-ins inte ska behöva känna till detaljer om innanmätet.

Istället ska olika plug-ins samverka och integreras via väldefinierade teknikgränssnitt såsom anrop eller meddelanden. Dessa teknikgränssnitt ska följa etablerade standarder, för att skapa hög interoperabilitet. Här är arkitekturen starkt influerad av Sambruks tidigare projektområde ÖTP (Öppen Teknisk Plattform).

Integrationen mellan någon plug-in och Medborgarnavet måste troligen kunna ske på flera alternativa sätt. Pragmatiskt kommer det därmed att uppstå viss varierande grad av lös koppling.

Det finns ett stort antal befintliga e-tjänster redan och vi kan inte förvänta oss att dessa på kort tid skulle göra en skräddarsydd integreration med Medborgarnavet, framförallt om det skulle ske via en mycket tätt kopplad arkitektur där en mängd teknikdetaljer måste uppfyllas av integrationsparterna.

Denna löst kopplade integration förväntas på lägsta nivån ske via interoperabla API:er, vanligen webservices. I vissa fall kan vi behöva anpassa oss till existerande äldre webservices utförda med SOAP och XML (eller t o m icke-interoperabla API:er via någon form av inkapsling), men det vanligaste tros bli pragmatiska REST-API:er med JSON. Olika sorters meddelandeförmedling eller till och med filkommunikation kan också tänkas.

Vi ansätter alltså följande samspel mellan plug-ins sinsemellan och med Medborgarnavet i sig:

- Interoperabla web services används för all datakommunikation mellan delarna.
- Delar kan inte förutsätta vilken teknikmiljö andra delar exekverar i utan det ska gå att blanda.
- Inga atomära transaktioner (ACID) ska förekomma där flera delar måste utföra uppdatering. Skulle sammansatta uppdateringar behövas (senare etapp) får de samordnas asynkront genom exempelvis Apache Kafka och BASE-mönster.
- Betoning av ansvarsområden, "black boxes", vad gäller de olika delarna, så att utarbetade förvaltningsavtal kan träffas och spelreglerna blir tydliga. Således ska det kunna vara helt olika ansvarande parter för Medborgarnavet i sig, e-tjänster, infokällor och infovisningsmoduler.

En viss mängd redirects mellan deltagande webbanvändargränssnitt förutspås även, vilket kräver ordentlig eftertanke med tanke på lös koppling.

Vi bör också ha en möjlighet att successivt addera plug-in-tekniker för att följa med teknikutvecklingen och för att förhandlingar med olika ägare och leverantörer av e-tjänster kan ge varierande utfall, troligen enligt maximen "det möjligas konst".

Å andra sidan är det inte bra om det blir en uppsjö av plug-in-tekniker att underhålla över tiden, exempelvis om varenda e-tjänst skulle integreras via varsin specifika teknik. Oanvända eller låganvända plug-in-tekniker behöver också successivt pensioneras. Vissa varianter som presenteras nedan kanske för övrigt aldrig behöver användas.

Vi har förhoppningen att det finns en win-win-möjlighet, så att ägare och leverantörer av e-tjänster känner att det finns en fördel i att integrera med Medborgarnavet – varvid det samtidigt uppstår en fördel för Medborgarnavet i och med många anslutna e-tjänster.

Nedan föreslås några olika plug-in-tekniker. Olika sorters partssamverkan, resonemang kring "lågt hängande frukter" och diverse initiativ får avgöra i vilken ordning plug-in-teknikerna implementeras (och vissa kanske alltså inte visar sig behövas alls, medan andra ännu inte är uttänkta).

CivicTech-projektet Medborgarnavet angav i sin Vinnova-ansökan att ansatsen är att försöka integrera med två pilot-e-tjänster, som naturligtvis bör prioriteras inom projektet: Valter (en e-tjänst för Gode män) samt E-ansökan Ekonomiskt Bistånd. Projektet har också tillfört en visnings-plug-in i form av Multifråga. Alla tre modulerna "ägs" inom projektdeltagaren Föreningen Sambruk.

3. E-tjänster som plug-ins

Nedan gås några alternativa plug-in-arkitekturer för e-tjänster igenom.

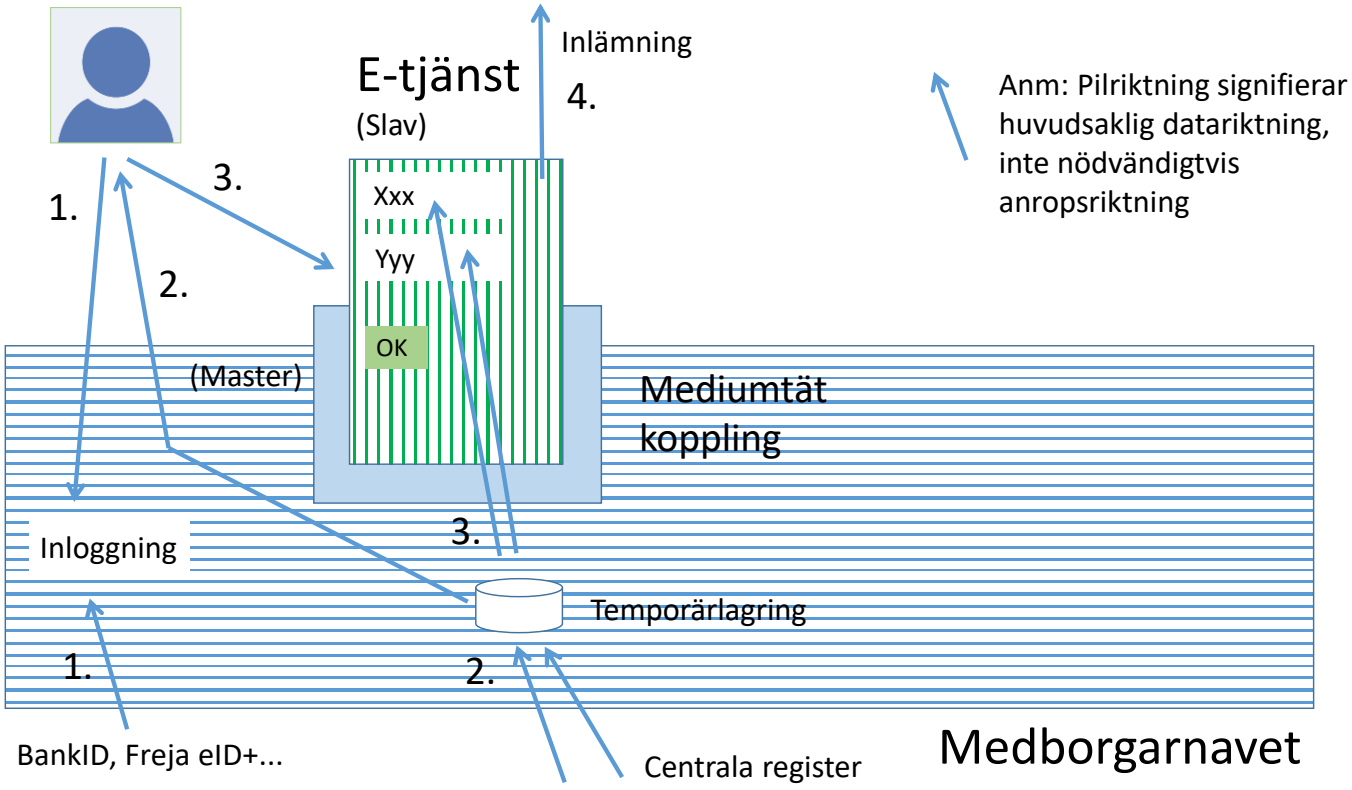
3.1. Plug-in-teknik: Mediumtät koppling med Medborgarnavet som master

En problematik med olika e-tjänste-plattformar, Mina Sidor-lösningar (det finns ett stort antal sådana), e-tjänste-ramverk osv är att "alla vill vara master". Till exempel: En leverantör har en Mina Sidor-lösning i vilken det går att hantera e-tjänster, men endast om starten av e-tjänsten sker inom just denna Mina Sidor-lösning och endast om inloggningen först har gjorts inom denna Mina Sidor. Dvs denna Mina Sidor måste vara master.

Ibland har man menat att federerad inloggning (t ex m.h.a. standarden SAML2) skulle lösa upp hela detta beroende, men inloggningen är endast en del i denna ekvation.

På motsvarande sätt är många e-tjänster utformade så att de endast kan startas inom en viss Mina Sidor-lösning e dyl, de har alltså ett alltför specifikt slav-beteende.

Emellertid är det fullt tänkbart att en viss e-tjänst utvecklas på ett sådant sätt att den kan startas från Medborgarportalen via dess plug-in-arkitektur, dvs att Medborgarportalen blir master:



Det finns flera alternativ inom detta scenario för plug-in-teknik. Ett kan vara sekvensen:

1. Användaren navigerar till Medborgarnavet.
   Inloggning sker inom Medborgarnavet.

2. Användarens egen info hämtas från centrala register.
   Infon kan ev visas via Medborgarnavet.

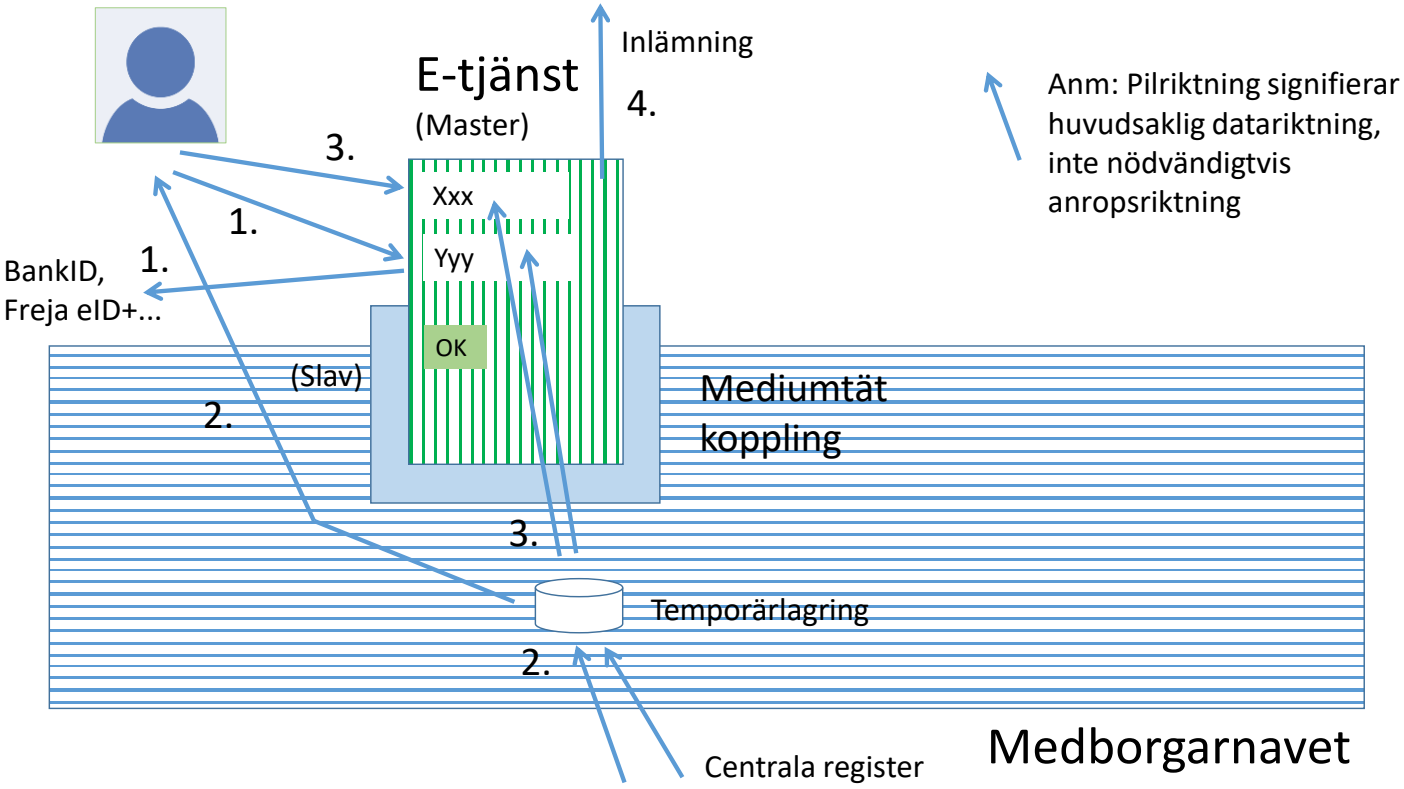
3. Medborgarnavet anropar e-tjänsten och överför info som kan passa i e-tjänstens fält.
   Användarkontrollen överförs till e-tjänsten.
   (Vissa fält är nu förifyllda men kan förmodligen ändras av användaren samtidigt som denne matar in övriga fält.)

4. Användaren klickar OK e dyl och utför därmed godkännande och inlämning till kommunen eller myndigheten, vilket helt hanteras av e-tjänsten. "Äkta" e-underskrift kan ev även ingå.

Att överföra kontrollen till e-tjänsten (punkt 2) kan ske på flera sätt; ett anrop kan intyga inloggningen, en "token" kan överföras, federering kan ske enligt SAML2, etc.

3.2. Plug-in-teknik: Mediumtät koppling med e-tjänsten som master

I de fall som e-tjänsten eller dess ev tillhörande Mina Sidor behöver vara master så sköter den/de om flera av stegen i skissen i föregående kapitel.



Det finns flera alternativ även inom detta scenario för plug-in-teknik. Ett kan vara sekvensen:

1. Användaren navigerar till e-tjänsten eller dess Mina Sidor.
   Inloggning sker där.

2. Användarkontrollen överförs till Medborgarnavet.
   Användarens egen info hämtas från centrala register.
   Infon kan visas i Medborgarnavet.

3. Användarkontrollen överförs tillbaka till e-tjänsten.
   E-tjänsten anropar Medborgarnavet och hämtar info som kan passa i e-tjänstens fält.
   (Vissa fält är nu förifyllda men kan förmodligen ändras av användaren samtidigt som denne matar in övriga fält.)

4. Användaren klickar OK e dyl och utför därmed godkännande och inlämning till kommunen eller myndigheten, vilket helt hanteras av e-tjänsten. "Äkta" e-underskrift kan ev även ingå.

Att överföra kontrollen till Medborgarnavet (punkt 2) kan såsom i första scenariot ske på flera sätt; ett anrop kan intyga inloggningen, en "token" kan överföras, federering kan ske enligt SAML2, etc.

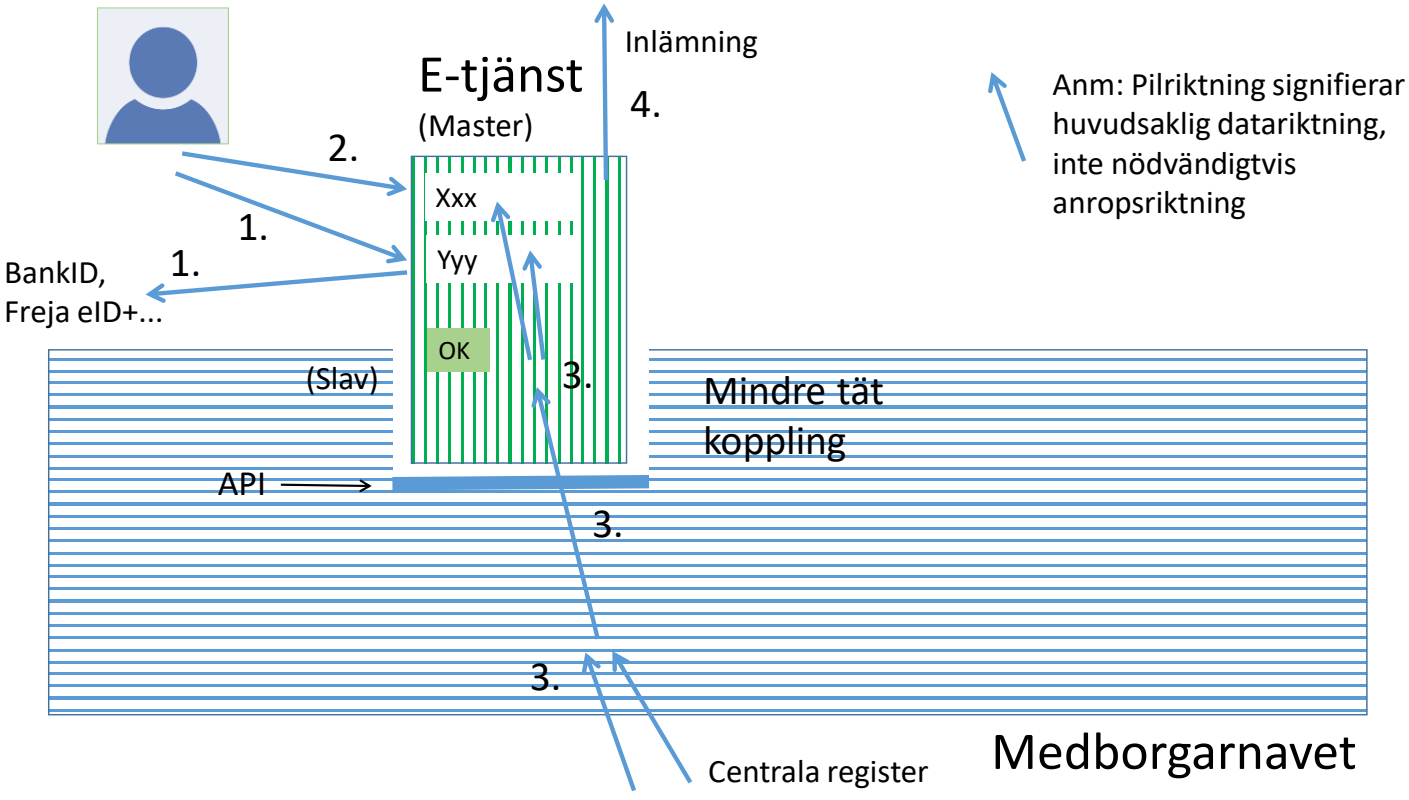
3.3. Plug-in-teknik: Mindre tät koppling via API med e-tjänsten som master

Här visar ett exempel på scenario för plug-in-teknik där e-tjänsten hela tiden har direktkontakt med användaren. Medborgarnavets användargränssnitt utnyttjas inte, utan Medborgarnavet är en ren slav och anropas via ett API.

En fördel med en mindre tät koppling av detta slag är att antalet teknikdetaljer som måste överenskommas blir betydligt färre, komplexiteten lägre och användaren slipper överflyttas mellan olika webbgränssnitt som sannolikt har skillnader i "look-and-feel". Hanteringen av diverse felsituationer blir också enklare.

En nackdel är dock att användaren inte lätt kan titta på alla sina data från centrala register. I så fall skulle det istället behövas en kombination av scenariodelar från ovanstående kapitel, eller att användaren får logga in separat i Medborgarportalen (eller ev kan ges SSO, Single SignOn, dit via federering/SAML2 e dyl).

En annan nackdel är att Medborgarnavet t ex i PSD2-användningsfallet kan behöva förnya medgivandeprocessen, där ställföreträdaren ska signera att det är OK att Medborgarnavet är Third Party Provider å dennes vägnar, mot banken ifråga. Detta torde kräva att användaren direktinteragerar med Medborgarnavet.



Sekvensen kan således vara exempelvis:

1. Användaren navigerar till e-tjänsten eller dess Mina Sidor.
   Inloggning sker där.

2. Användarkontrollen behålls i e-tjänsten och användaren kan börja fylla i fält etc.

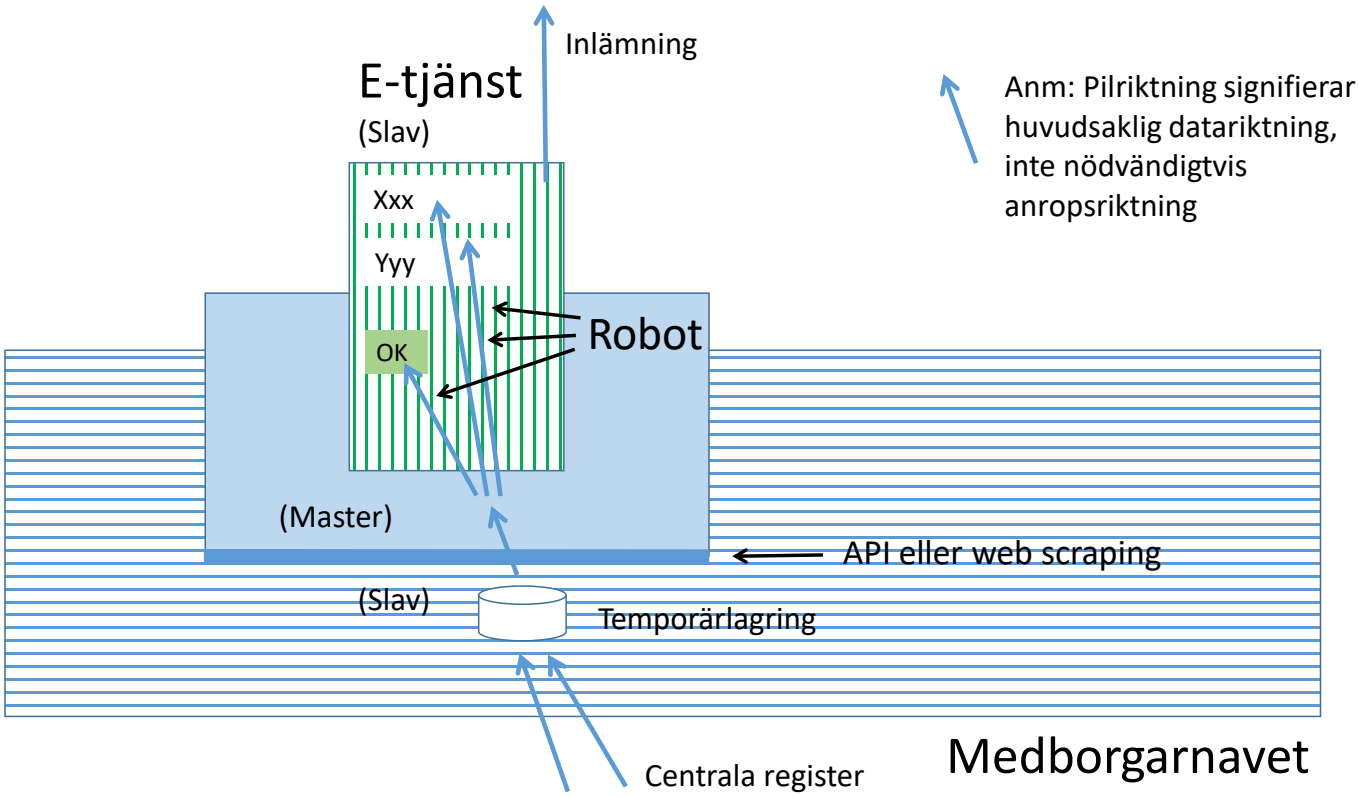
3. Användaren kan trycka på en knapp i e-tjänsten för infohämtning.
   E-tjänsten anropar då Medborgarnavet via API:et och ber om info som passar till e-tjänstens fält.
   Medborgarnavet i sin tur anropar de centrala registren och svarar till e-tjänsten med begärda svarsfält.
   (Vissa fält blir nu förifyllda men kan förmodligen ändras av användaren samtidigt som denne matar in övriga fält.)

4. Användaren klickar OK e dyl och utför därmed godkännande och inlämning till kommunen eller myndigheten, vilket helt hanteras av e-tjänsten. "Äkta" e-underskrift kan ev även ingå.

Även en omvänd variant där Medborgarnavet är master kan tänkas, men är möjligen mindre trolig.

3.4. Plug-in-teknik: Robot

Robotar (RPA, Robotic Process Automation) får kanske anses vara en slags reservlösning. Man kan behöva ta till en robot för att lösa integrationen om man har e-tjänster som är instängda och inte alls går att integrera på normala sätt, eller om en ägare/leverantör är ointresserad av normal integration, eller om normala integrationskostnader inte går att motivera.



En robot i detta sammanhang är en mjukvara som på ett strukturerat sätt försöker efterlikna en användares logiksinne, ögon och händer. Roboten är "master" och kan identifiera fält i en e-tjänst, röra pekaren, trycka på knappar samt fylla i fält. Vissa robotar klarar av webbteknik, detta kallas oftast "web scraping". Andra klarar även s.k. native-applikationer (specifika för Windows, Mac, Linux osv). Eftersom vi siktar på e-tjänster via Internet torde det vara web scraping som är aktuellt i vårt fall.

Robotprojekt pågår i många kommuner och en av de starkaste anledningarna är just att kommunernas standardapplikationer är inlåsta och att andra integrationsmöjligheter saknas. Men roboten har också fördelen att kunna programmeras för att känna till handläggningsregler, lagar samt bedömnings-kriterier. Så kan rutinhandläggning automatiseras (över applikationsgränser), de enkla rutinärendena gå snabbt och endast de komplexa ärendena kräva mer omfattande handläggarinsats.

En del anser att trenden med artificiell intelligens (AI) kommer att göra robotarna så intelligenta att en mycket stor andel av handläggningarna kommer att kunna skötas. Dock bör man vara försiktig, det finns även gott om AI-bakslag.

Det finns även många nackdelar med robotar. E-tjänstens utseende kanske ändras varvid roboten kan gå helt vilse och till och med fylla i information som valideras korrekt men kan ge fullständigt galen

verkan. Robotprogrammering kan kosta en del, liksom att robotmjukvaran kan ha en dyr licens. Robotar kan också konservera "ihoptejpade" lösningar – incitamentet att verkligen få en e-tjänste-leverantör att acceptera en mer kvalitetssäkrad integration än via robot, kan minska.

Icke desto mindre är ofta robot-teknik det enda som står till buds.

I skissen har vi visat att robotens gränssnitt mot Medborgarportalen kan vara antingen via API eller via web scraping. Då så är möjligt bör förstås API föredras pga ovan nämnda kvalitetsrisker med web scraping.

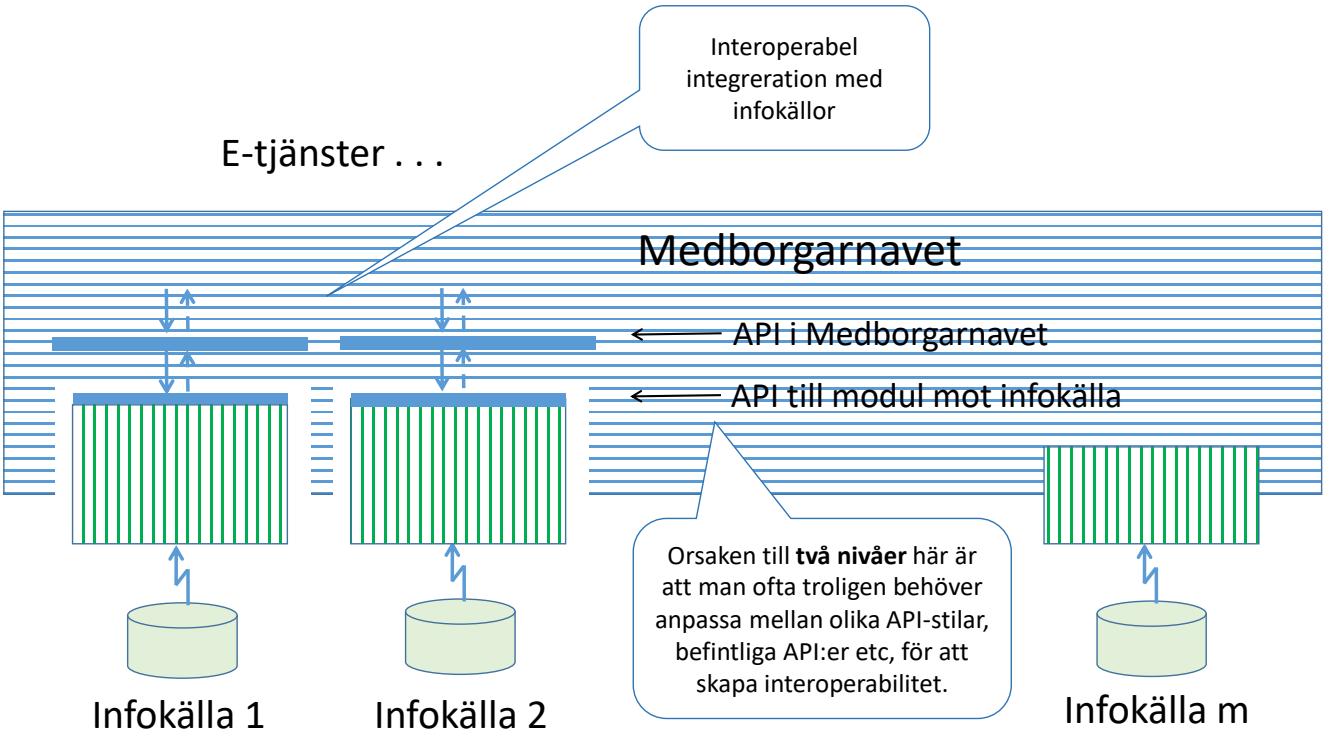
Vi har dock förutsatt att robotens gränssnitt mot e-tjänsten är via web scraping eftersom detta troligen skulle vara huvudorsaken till att en robot alls används i detta fall – att e-tjänsten ingen API-möjlighet har (men om den har det, bör API-scenariot föredras istället för robot).

Användningssekvensen kan vara någon kombination av de sekvenser som exemplifieras i ovanstående scenariokapitel, beroende bl a på e-tjänst och robot ifråga.

4. Infokällor som plug-ins

Nedan gås några alternativa plug-in-arkitekturer för informationslämnande tjänster igenom.

En grundprincip är att infokällorna kan tänkas vara skapade med helt olika teknik, liksom vad gäller ipluggbara e-tjänster, varför en "lös koppling" via interoperabla gränssnitt behövs.

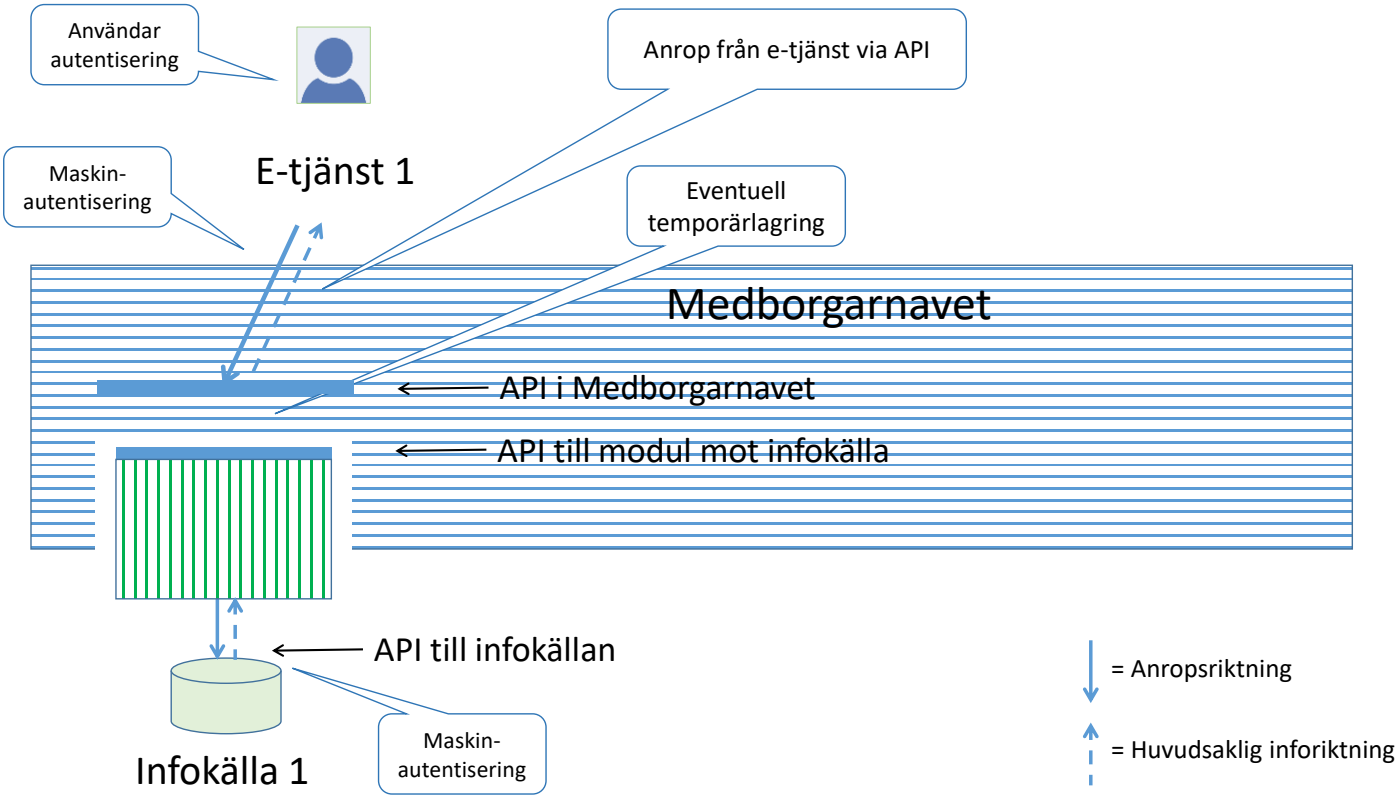


De två API-nivåer som nämns i figuren ovan kan eventuellt i enklare sammanhang sammanfalla. I andra fall kan man tänkas vilja skapa ett (1) generaliserat API i Medborgarnavet mot flera olika API:er mot olika infokällor – en slags enkel generaliserad (s.k. kanonisk) API-nivå för inkapsling.

4.1. Plug-in-teknik: Enkel kaskadkoppling mot infokälla

Med enkel kaskadkoppling förstås här att ett synkront anrop helt enkelt kan skickas vidare, och svaret likaså förmedlas tillbaka, i en synkron anropskedja.

När detta förfarande går att åstadkomma blir det enkelt eftersom ingen "state-information" behöver vidmakthållas och synkroniseras. Det enda som behöver persisteras är loggning, samt ev temporärlagring av rådata för efterföljande visning.



Med denna princip kan man ganska enkelt successivt anropa vidare genom skikten. Alla anropen är synkrona, så de väntar tills svar erhålles (inklusive om felsvar skulle uppstå), vilket förenklar framförallt felhanteringen.

Ett exempel för infokälla här är SKR:s SSBTEK (Sveriges Kommuner och Regioners Sammansatt Bastjänst Ekonomiskt Bistånd). Detta API kräver maskinautentisering (vitlistat kommun-cert osv) och en överenskommelse om "syfte" mm.

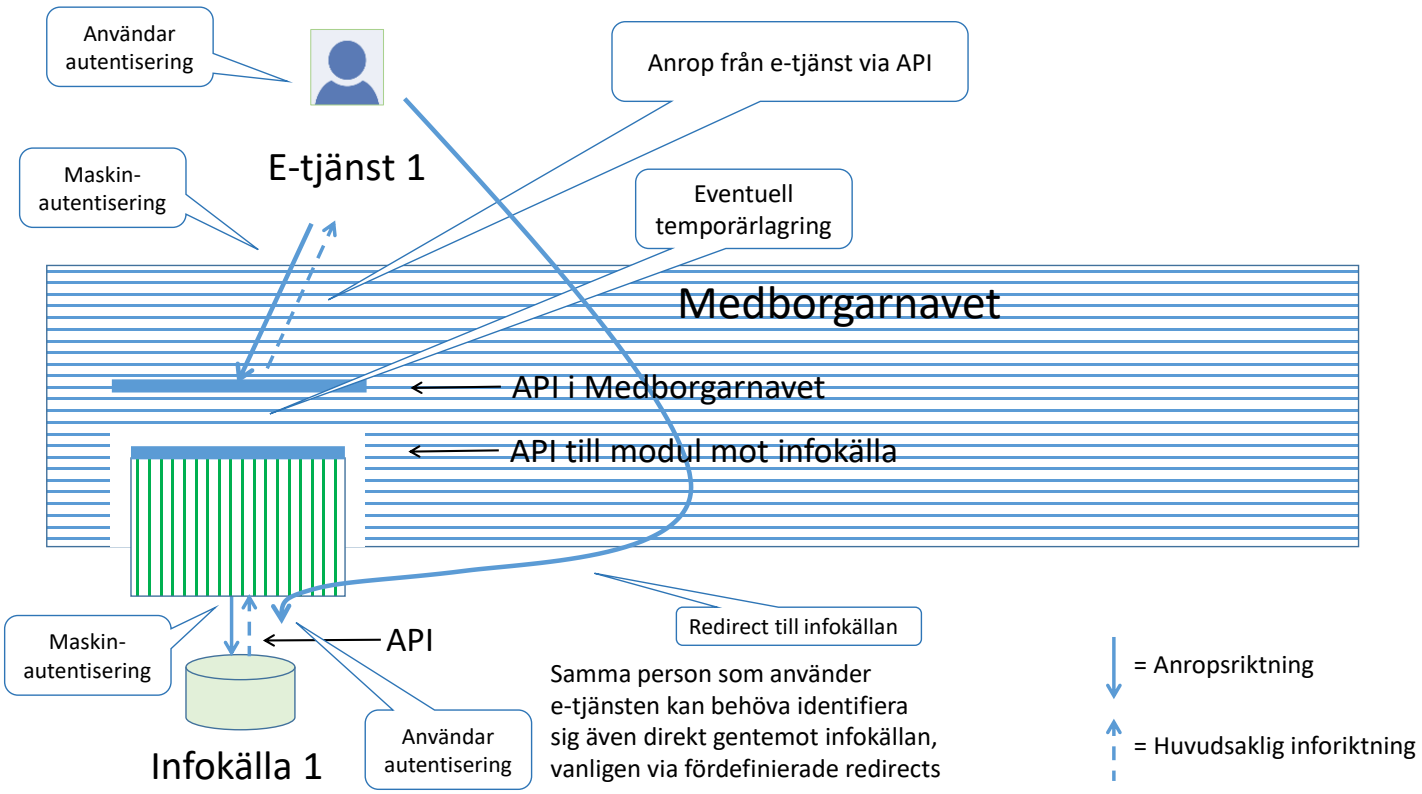
E-tjänsten måste tillika maskin-autentisera sig mot Medborgarnavet, vilket kan ske på flera sätt; ett anrop kan intyga inloggningen, en "token" kan överföras, federering kan ev ske enligt SAML2, etc.

E-tjänsten måste utlova att den i sin tur har autentiserat en användare med adekvat säkerhet.

Om man önskar att användaren ska kunna se och/eller bearbeta datat inom Medborgarnavet, krävs även viss redirecthantering.

4.2. Plug-in-teknik: Komplex kaskadkoppling mot infokälla

Med komplex kaskadkoppling förstås här att ett en sammansatt sekvens av exempelvis anrop och web-redirects kan krävas för att få datautybte med infokällan till stånd.



För denna princip är samverkan mellan skikten mer komplicerad och det förekommer en blandning av API-anrop (synkront och/eller asynkront slag) och redirects via personens webbläsare så att infokällans krav kan tillfredsställas.

Ett exempel här är bankernas API:er enligt PSD2-lagen. Medborgarnavet tar här rollen som TPP (Third Party Provider (en roll som mer generellt ofta beskrivs som en TTP, Trusted Third Party) gentemot de andra två parterna användare respektive bank.

För att användaren ska kunna ge samtycke till banken om att Medborgarnavet/TPP ska få tillgång till bankkontoinfo, måste bl a användarens webbläsare få kontrollen via en sekvens av redirects, inklusive interaktion med BankID.

Förhoppningen är att Medborgarnavet ska kunna abstrahera bort en hel del av denna komplexitet så att e-tjänsten kan uppleva ett mycket enklare API. Emellertid krävs alltså vissa redirects från/till e-tjänsten.

E-tjänsten måste utlova att den i sin tur har autentiserat en användare med adekvat säkerhet.

En användning av denna plug-in-variant finns t ex betydligt mer detaljerat beskriven i Medborgarnavet_PlugInArk_Valter_v2021-06-07 (eller nyare).

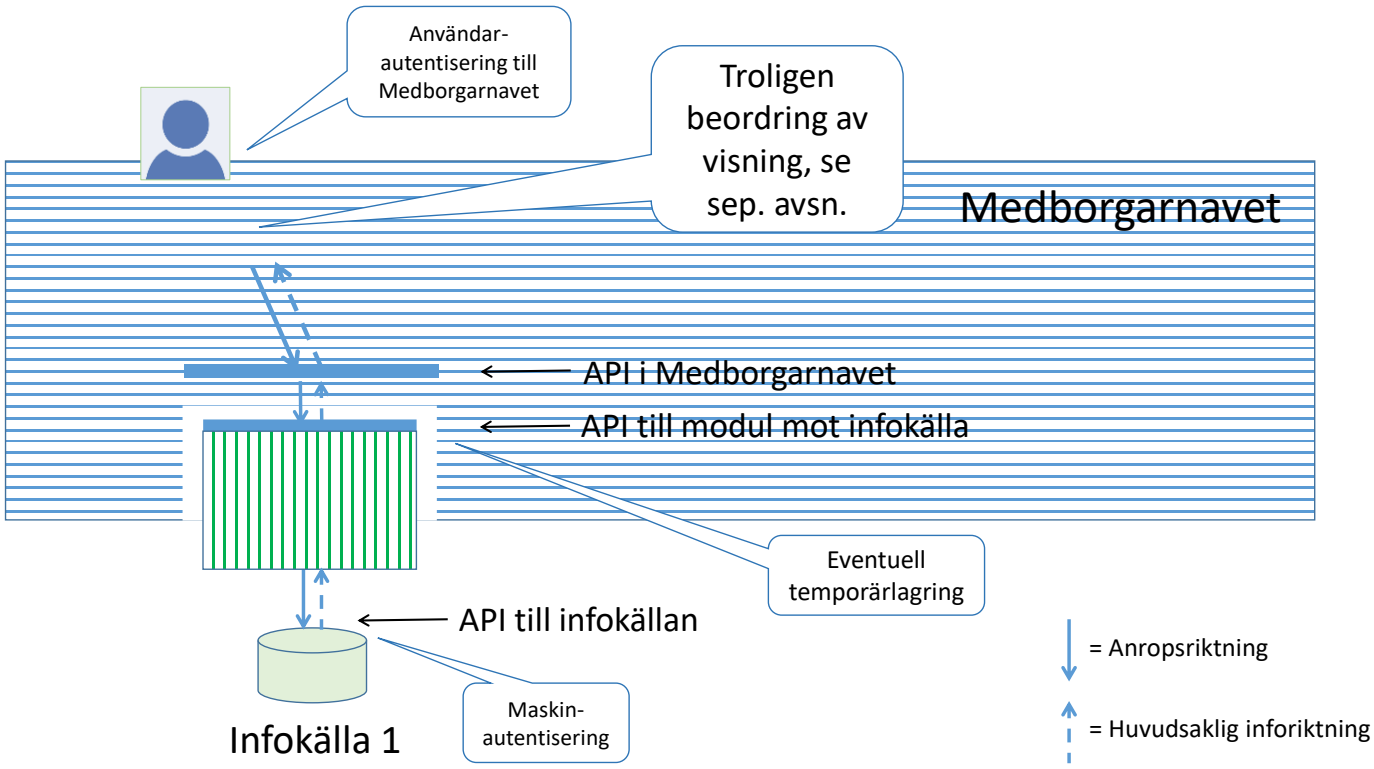
4.3. Plug-in-teknik: Direktinloggning i Medborgarnavet och koppling mot infokälla

Denna variant kommer ifråga när en användare loggar in direkt i Medborgarnavet och inte går via någon e-tjänst.

Anledningen kan exempelvis vara att specifikt visa information om sig själv, se vidare i avsnittet *5. Visningsmoduler som plug-ins.*

Med koppling förstås här ett synkront anrop till en modul mot en infokälla (eller flera).

Det som behöver persisteras är loggning, samt ev temporärlagring av rådata för visning.



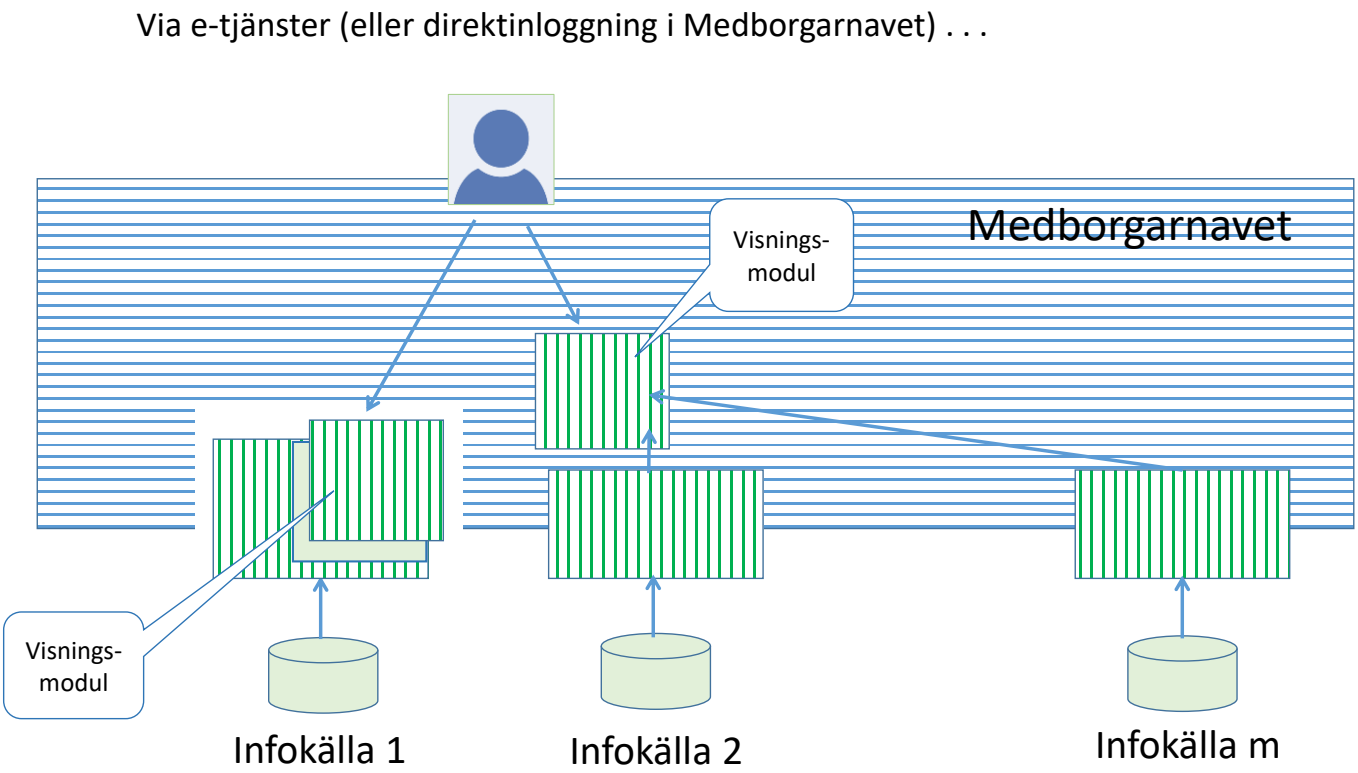
Ett exempel för infokälla är även här SKR:s SSBTEK (Sveriges Kommuner och Regioners Sammansatt Bastjänst Ekonomiskt Bistånd), se ovan.

Användaren måste logga in i Medborgarnavet med adekvat autentisering.

5. Visningsmoduler som plug-ins

I vissa fall kan man tänka sig att temporärlagring och visning kan ske helt inom Medborgarnavet, men då bör dessa moduler tydligt avgränsas som plug-ins.

Det kan troligen bli vanligt att den part som har kunskapen om den specifika infokällan även anordnar en visningsmodul för just denna typ av data, dvs en slags "black box-ansvar" för en viss typ av information. Då inkluderas både hämtning och visning. Man kan tänkas basera sig på befintliga moduler och då kan man få acceptera att hämtning och visning inte är separerade, se subkapitlen nedan.

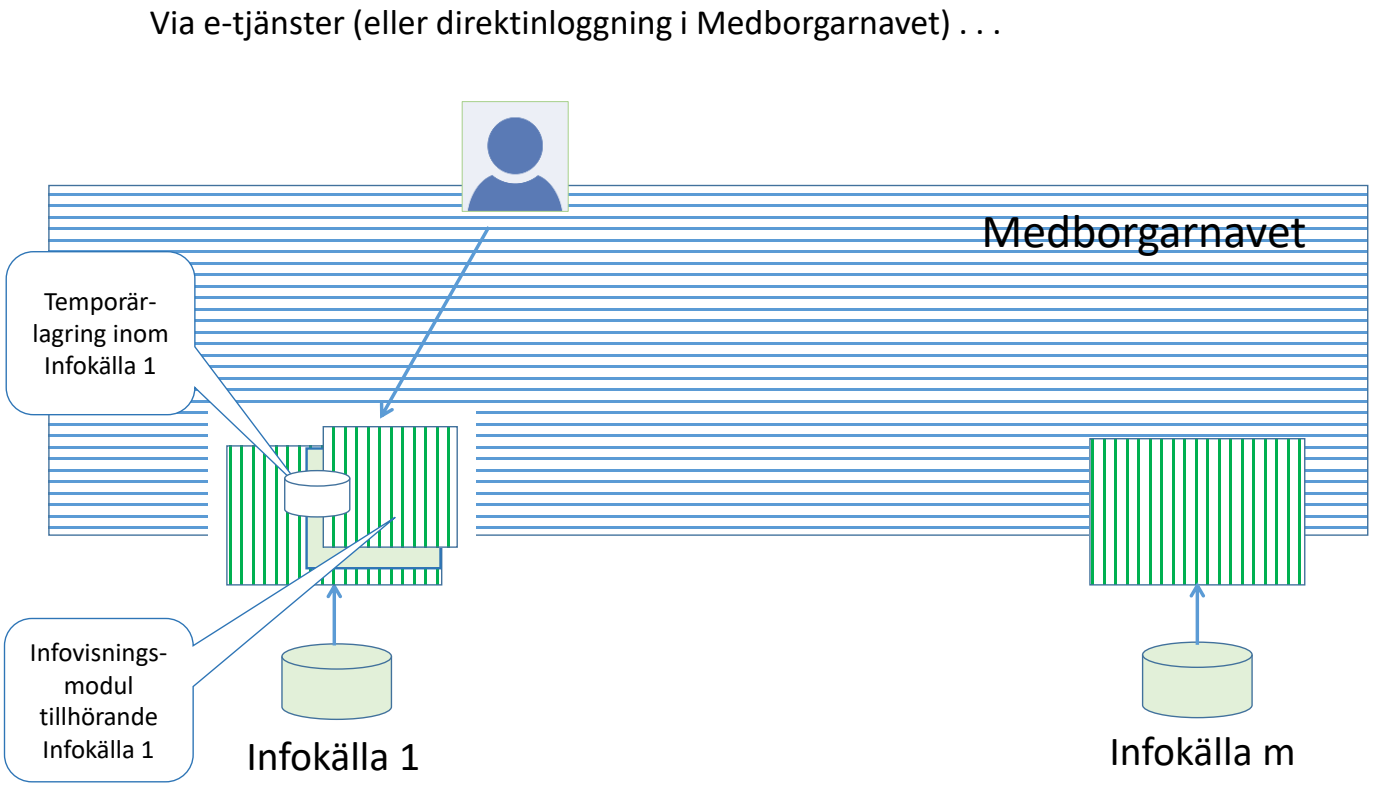


I plug-in-figuren ovan är alltså indikerat att infohämtningsmodulen till infokälla 1 också innehåller en visningsmodul.

För infokälla 2 däremot existerar en visningsmodul som är separat från infohämtningen.

5.1. Plug-in-teknik: Integration mot visningsmodul inom en infokällmodul

Om det exempelvis finns en befintlig infovisning inom en informationshämtningsmodul eller om man föredrar det av andra skäl, hamnar visningen inom "black box-ansvaret" för en viss informations-plug-in.



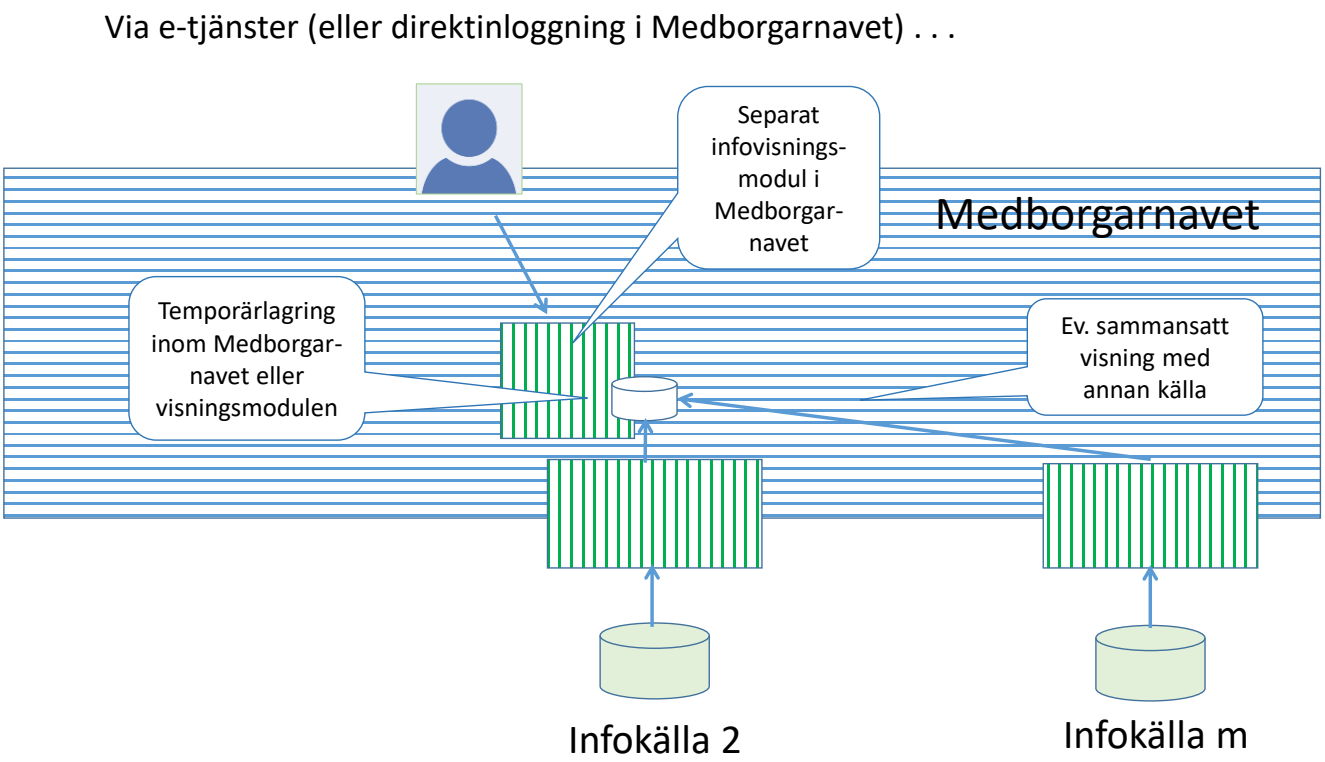
Sambruks Multifråga är av denna typ. Den har i sin ursprungsversion både hämtningslogik för SSBTEK, visningslogik och temporärlagring.

En fördel med varianten är att ansvaret för informationen blir väldigt tydligt och att det är troligt att versionering av själva informationsstrukturen över tiden, och underhållet av visningsmodulen, blir i synk. Detta torde också vara den billigaste lösningen.

En nackdel är dock att om användaren navigerar runt bland olika visningsmoduler så kommer de att ha mycket olika användargränssnittsstilar. Man kan dock se att detta ibland tycks accepteras utan alltför mycket knot, bara själva informationen i sig är bra, ta exemplet Skatteverkets webbar där deklarationsdelen sedan länge har en helt annan användargränssnittsstil än Skattekontodelen.

5.2. Plug-in-teknik: Integration mot separat visningsmodul inom Medborgarnavet

En "renare" variant är att separera infovisningen från infokällan.



På detta sätt går det att renodla visningen så den får ett konsekvent utseende.

Beroende på hur mycket vi hinner i nämnt Vinnovaprojekt, kan man tänka sig att separera ut Multifrågas visningsdel till en sådan här separat infovisning.

Olika sammanställningar kan också skapas å medborgarens vägnar, baserat på temporärlagrat data från flera källor.

En del andra för- och nackdelar framgår av föregående avsnitt, fast tvärsom.

Datalagringssymbolen är i figuren medvetet ritad över gränsen mellan visningsmodulen och själva Medborgarnavet – här kan man tänka sig olika varianter beroende på förutsättningarna.

Ytterligare varianter finns, exempelvis att temporärlagringen helt ligger inom infokällemodulen, medan infovisningsmodulen får hämta data vid visning, genom ett interoperabelt API.

6. Ansvarsgränser

Det blir mycket viktigt att mejsla ut ansvarsområden för olika aktörer i samband med Medborgarportalen.

Projektets ansats är enligt Vinnova-ansökan att efterlikna bankvärldens PSD2 (EU-lag) genom att Medborgarportalen skulle utgöra en s.k. Third Party Provider. Alltså behöver både användaren och centrala register lita på Medborgarportalen – denna ska utföra informationshämtning å användarens vägnar.

Men vilken roll får då e-tjänster (och ev Mina Sidor) i ovanstående scenarion? Tydliga och konkreta avtal behöver tecknas med den part som har ansvaret för e-tjänsten, dvs vanligen en kommun eller myndighet.

Innan avtal kan träffas måste en självdeklaration skrivas av plug-in-ansvarig. Förvaltningspersonalen för Medborgarnavet måste göra en viss inspektion av plug-in-modulen för att säkerställa att den håller tillräcklig säkerhetsklass.

Många gånger ska det avtalas att Medborgarnavet ska kunna lita på en personautentisering gjord i e-tjänsten.

Avtalen behöver ange villkoren för att Medborgarportalen ska "våga godkänna" att dess status som Third Party Provider"ärvs" av e-tjänsten. Att ses som Trusted innebär förstås ett synnerligen stort ansvar.

Två områden som måste ingå i avtal/självdeklaration/inspektion är nivå enligt Tillgänglighetsdirektivet, samt GDPR-relaterade frågor/dokument.

Kommunen eller myndigheten måste i sin tur avkrävas att ha avtal på plats där den mjukvaruleverantör av e-tjänst som kommer ifråga avkrävs villkor för att ärvandet är godtagbart. Detta behövs förstås inte om kommunen/myndigheten själv utvecklat e-tjänsten. Om kommunen/myndigheten inte själv driftar e-tjänsten måste motsvarande avtalsvillkor vara på plats även gentemot driftpartnern. Samt i sin tur dennas ev underleverantörer, osv.

Således kan avtalskedjan tänkas bli:

> Både **Användare** och **ansvarig för centrala register** (1st & 2nd Party)
> > **Medborgarnavet** (3rd Party)
> > > **Kommunen/myndigheten** som använder e-tjänsten
> > > > **Mjukvaruleverantör** av e-tjänsten
> > > > > Ev **driftleverantör** av e-tjänsten
> > > > > > Ev **IT-underleverantörer**...

Ovanstående avtalsresonemang kan upplevas utgöra "kaskadkopplingen" för typiska personuppgifts-biträdesavtal enligt GDPR, men vi har hittills ansatsen att ovanstående behöver vara specifika tjänsteavtal och att det alltså dessutom behövs separata personuppgiftsbiträdesavtal, eftersom dessa i grunden har ett annat fokus (personuppgiftsskydd istället för reglering av ansvar vid Trust). Möjligen skulle detta dock gå att sammanföra.

Se även om "black boxes" för plug-ins etc i kapitlet *2.4. Hur tät koppling via* plug-in-tekniken?

7. Säkerhet

Nedan nämns några korta förutsättningar vad gäller säkerhet. I varje praktiskt fall behöver dessa frågor mejslas ut ytterligare.

Säkerhetsnivån hos Medborgarnavet måste vara mycket hög. Adekvata tekniska och administrativa åtgärder ska vidtas beträffande säkerheten. Vissa informationstyper kan vara öppen information av olika slag, men vi baserar mycket av vår erfarenhet från scenarios där hög sekretess råder, såsom inom socialtjänstesektorn.

Man måste även beakta GDPR. Dock behöver alltid en sådan lag som GDPR balanseras mot andra krav, i stil med följande citat ur GDPR: "...Med beaktande av den senaste utvecklingen, genomförandekostnader och behandlingens art, omfattning, sammanhang och ändamål samt riskerna...".

Utformning av systemutvecklingsmoduler, API:er och datalagring måste ha hög driftsmässig isolation då sådan isolation ska finnas. Likaså måste isolation mellan olika "tennants" i en eventuell "multi-tennant-driftning" var mycket strikt.

Autentisering måste göras enligt den kravbild som respektive plug-in kräver, vilket ofta blir på nivån stark autentisering med exempelvis BankID eller Freja eID+.

8. Äkthetsintyg för informationsfält

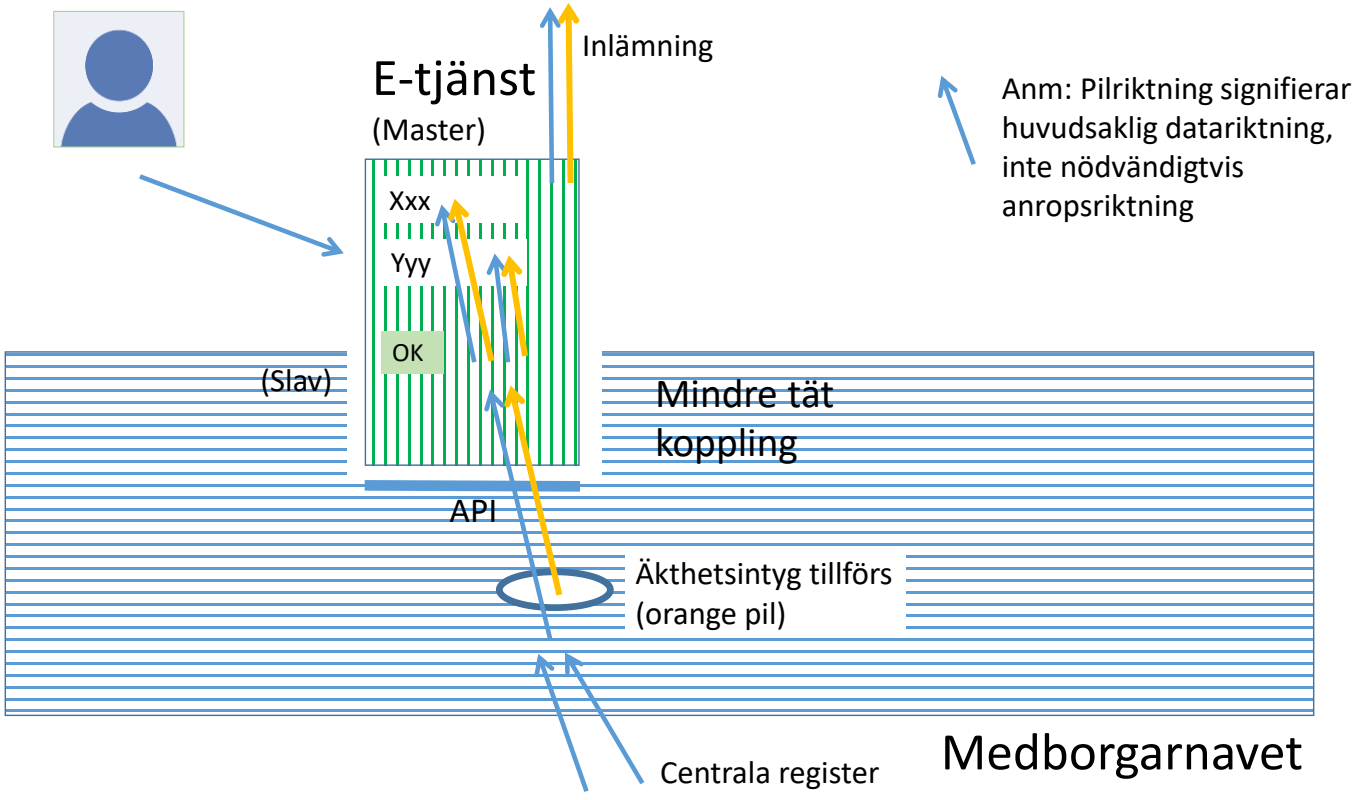
En intressant potential är att underlätta en handläggares bedömning av exempelvis en inkommen ansökan.

I de fall då information hämtas direkt från centrala register, för att sedan utan modifiering överföras till fält i en e-ansökan och sedan lämnas in, bör handläggaren i de allra flesta fall kunna lita på att den centrala registerinformationen är korrekt varför denne inte behöver kolla upp dessa fält ännu en gång, senare i handläggningen.

Många gånger tänker man sig i de här sammanhangen att informationen i förifyllda fält granskas av användaren och ändras om den inte skulle stämma. Sedan kan användaren intyga korrektheten med en knapp för "Godkänn och lämna in", eller via en "äkta e-underskrift". En förebild som ofta nämns är Skatteverkets deklarationsblankett med förifyllda inkomstbelopp men samtidigt med möjlighet för användaren att komplettera varje fält med vad användaren anser är mer korrekt.

Medborgarportalen skulle parallellt kunna sigillera informationen som kommer från de centrala registren, med hjälp av kända kryptografiska standarder. Detta blir alltså en slags äkthetsintyg. Om sådana äkthetsintyg forslas vidare genom e-tjänsten och till den bedömande handläggarens IT-stöd så kan alltså handläggaren slippa ett antal verifierande arbetsuppgifter. Tekniken bakom sigilleringen gör att eventuell förvanskning av fälten ifråga går att automatiskt upptäcka i efterhand. Självklart ska inte fält som användaren manuellt korrigerat kontrolleras mot äkthetsintyget, det skulle förstås misslyckas, användarkorrigerade fält måste få en markering så handläggaren vet.

Följande skiss lånad från API-scenariot ovan visar var äkthetsintyget kunde skapas:



Ett exempel: Sambruk har stor erfarenhet av ansökningar om bistånd inom socialsektorn. En stor arbetsbörda läggs idag på bedömande handläggare för att verifiera att dessa ifyllda uppgifter stämmer. Dels är informationen svår för den sökande att hålla reda på så det blir rätt, dels förekommer bedrägeriförsök. Fysiska kvitton och bankkontoutdrag ska äkthetsbedömas. I vissa fall måste handläggaren ringa till centrala myndigheter såsom Försäkringskassan. I förhoppningsvis många fall finns det informationstjänster som handläggaren kan använda för kontrollen, såsom Sambruks Multifråga (som via det s.k. SSBTEK-API:et hämtar från åtta myndigheter), men detta är dock ändå ett tidskrävande kontrollsteg. Tekniskt bifogade äkthetsintyg enligt principen ovan skulle kunna underlätta avsevärt för socialhandläggarna så att resurser frigörs för kurativt arbete istället. På detta sätt skulle man följa principen om att "skapa hög datakvalitet redan vid källan", istället för att försöka rätta senare.

Även automatiserat beslutsfattande (via robot eller "vanlig programmering") torde ha nytta av information med äkthetsintyg.

Man bör dock beakta att det i något fall funnits juridiska invändningar kring att myndighetsinformation har förts vidare på liknande sätt som ovan.

9. Sambruks referensarkitektur

För att ge bakåtreferenser och spårbarhet samt även peka på erfarenheter och eventuella framtida risker med Medborgarnavet inkluderas här några korta noteringar angående ovanstående kapitel innehållande plug-in-scenarion, relativt de mönster som beskrivs i Sambruks Öppen Teknisk Plattform (ÖTP).

ÖTP var ett projekt som tidigare pågick under ett antal år inom Sambruk. ÖTP kom att användas i olika sammanhang, framförallt kanske som s.k. kravmaster och referens i vissa kommunala offentliga upphandlingar.

Mönstren i ÖTP är influerade av dåtidens resonemang kring SOA, Service Oriented Architecture. Även om generellt sett förväntningarna på SOA får sägas ha varit för stora och besvikelser ibland uppstod, är det många delar av SOA-mönstren som ändå får anses vara "best practice" idag, men ofta under beteckningar som API Orientation och i vissa bemärkelser Micro Services.

Några anledningar till att ÖTP ändå inte kom att ge så stort resultat som vi då hoppades, var:

1. Kommuner har sällan råd att skräddarsy applikationer enligt egen specifikation, de köper standardapplikationer.

2. Leverantörerna av dessa standardapplikationer vill helst hålla användningen helt inom sin applikation, de har haft få incitament att skapa integrationsmöjligheter till omvärlden. I de fall som integrationsmöjligheter skapats, har ibland prissättningen varit oblyg.

3. Ett av de få tillfällen då kommunen har en större möjlighet att kräva integrerbarhet för sådana standardapplikationer är vid upphandlingstillfället. Man måste då ställa tydliga krav men samtidigt inte ställa så hårda krav att man inte får in några vettiga offerter alls, eller får höga priser – en tidvis besvärlig kompromiss.
   Icke desto mindre finns fall där upphandlingskrav som refererade öppenhet enligt ÖTP:s kravmaster faktiskt gjorde att integration lyckades utmärkt.

4. Ett helt annat problem är att få de ansvariga för centrala register att tillgängliggöra integrerbarhet , t ex via API:er. SSBTEK är dock ett gott exempel där mångårigt påverkansarbete från bl a Sambruk gav frukt, men det finns mängder av inlåsta centrala register där god medborgarinsyn inte beaktats.

Hur relaterar då ovanstående punkter till det nuvarande Medborgarnavsprojektet? Jo, problemet kan även idag vara att få leverantörerna av e-tjänster, Mina Sidor etc (som ju är en slags standardapplikationer) att införa integrerbarhet så att plug-in-scenariona ovan faktiskt går att implementera. Tänkbara sätt att tackla detta:

- Reservlösningen kan vara robotanvändning, se kapitlet ovan.

- Det föreliggande Medborgarnavsprojektet har också konkret "ägarmakt" över två "e-tjänste-standardapplikationer", Valter och eAnsökan EkBist (samt en visnings-standardapplikation, Multifråga) och vi hoppas att kunna anpassa dessa för önskad integrerbarhet enligt något av plug-in-scenariona ovan. Dessa kan tjäna som förebild.

- Förhoppningsvis har den allmänna API-trenden inom IT-världen nått leverantörerna av standardsystem i kommunvärlden, så att integrationströskeln blir lägre idag.

- Man bör argumentera för inkludering av integrerbarhetskrav i upphandlingsunderlag.
- Vad gäller API:er till centrala register bör man använda argument från MyData-rörelsen mm.

Några spårbarhetsreferenser till ÖTP-referensarkitekturen:

- Plug-in-interaktionen i scenariona ovan utgör mönster X och Y i avsnitt 3.2.1 i ÖTP, då e-tjänsten är master (det omvända finns dock inte lika tydligt med i ÖTP).
- Scenariot ovan med robot utgör mönstret i avsnitt 3.2.3 i ÖTP.
- Vad gäller centrala register utgör Medborgarnavet anpassningsskiktet för fall 1 och 2 i avsnitt 3.3 i ÖTP.

Se ÖTP v3.0 Referensarkitektur, via www.sambruk.se. I dagsläget på länken:
https://sambruk.se/wp-content/uploads/2019/02/Sambruk_OTP_RefArk_v3_0_2012-02-07.pdf.