Systemdokumentasjon

Elektronisk valgadministrasjon - EVA Skanning

VALG

Innhold

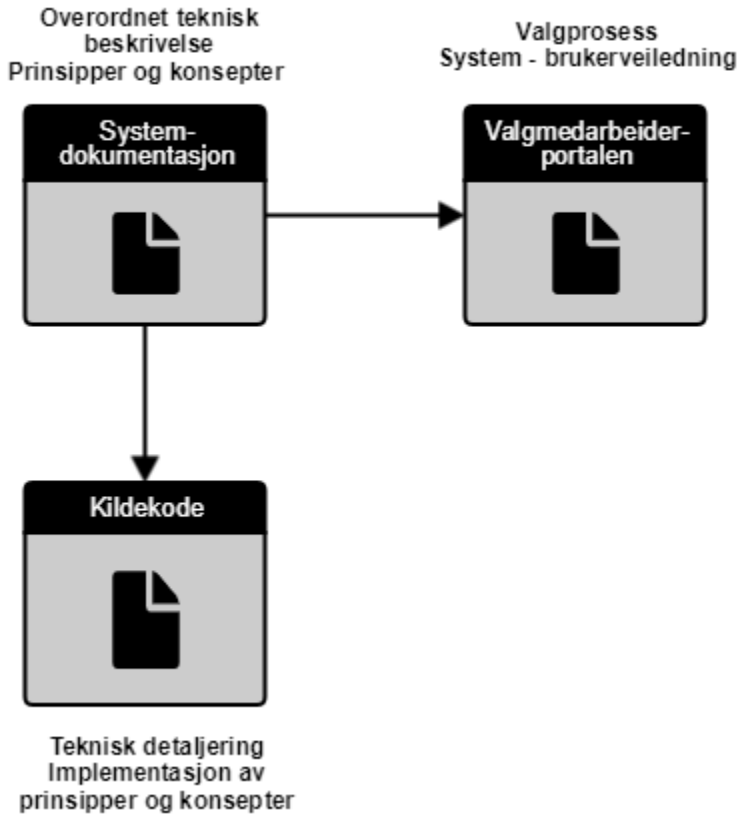
Introduksjon

Systemdokumentasjonen gir en overordnet beskrivelse av EVA -systemet fra et funksjonelt og teknisk perspektiv.

Dokumentet gir beskrivelser på et konseptuelt nivå, og er ikke en uttømmende teknisk beskrivelse av EVA. Konsepter og prinsipper gir i sin tur bakgrunn til kildekode og konfigurasjon som er hoveddokumentasjonen for systemet på et detaljert teknisk nivå.

Inngående beskrivelser av selve valgprosessen er gitt på valgmedarbeiderportalen, med brukerveiledning, prosess- og rutinebeskrivelser, samt skjematiske framstillinger av prosess.

Skisse som illustrerer hvordan dokumentasjonen er organisert:



Strategi for systemdokumentasjon

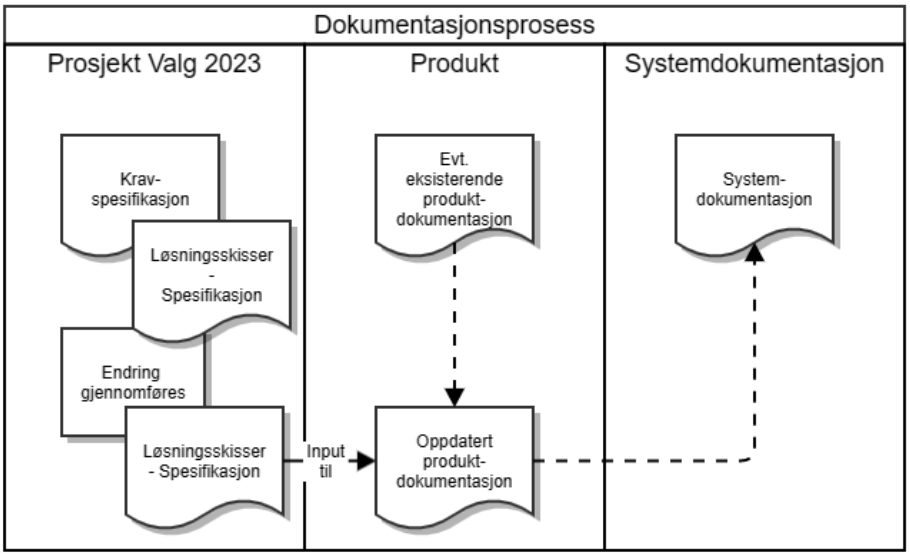
Valgdirektoratet bruker Atlassian Confluence (wiki) for å dokumentere IT-løsninger og systemer som utvikles og forvaltes av Valgdirektoratet. Systemdokumentasjon er derfor generert ut i fra det dette systemet.

Dokumentasjonen forankrer en enhetlig forståelse av hvordan systemet virker for å oppfylle de krav som stilles, dokumentasjonen brukes også som hjelpemiddel i utviklingsprosessen. Dokumentasjonen er ikke detaljert på et kodenært nivå, for å sikre en bestandig dokumentasjon som gir nødvendig overblikk og forståelse av prosesser og konsepter. Konkrete implementasjonsdetaljer må leses i kildekoden.

Effektiv systemdokumentasjonsforvaltning

Systemdokumentasjonen for Valgdirektoratets IT-systemer er bygget opp som et aggregat av de enkelte produktdokumentene, man unngår dermed duplisering og forvaltning av duplisert dokumentasjon og informasjon. Dette betyr også at informasjonen i systemdokumentasjonen er den samme informasjonen som brukes internt i Valgdirektoratet knyttet til systemforvaltningen, og således er *levende* og *benyttet* dokumentasjon.

Skissen illustrerer hvordan systemdokumentasjon bygges opp og vedlikeholdes



- Et krav stilles og spesifikasjon / løsningsskisser lages i Confluence.
- Endringen gjennomføres, systemet er nå endret ihht. spesifikasjon / løsningsskisse
- Eventuell eksisterende produktdokumentasjon for det gitte området / den gitte funksjonen oppdateres, eller ny dokumentasjon legges til
- Systemdokumentasjonen oppdateres siden den er et aggregat av produktdokumentasjonen

Overordnet systembeskrivelse - EVA

EVA - (Elektronisk valgadministrasjon) er Valgdirektoratets IT-støttesystemportefølje som kommuner og fylkeskommuner kan benytte seg av for å forenkle valggjennomføringen. EVA er ikke et saksbehandlingssystem og kan ikke erstatte valgstyrene sitt ansvar for å påse at valghendelsene blir gjennomført etter gjeldende regelverk, men fungerer som et støtteverktøy for kommuner og fylkeskommuner i de ulike fasene av valggjennomføring. EVA har funksjonalitet som støtter opp om gjennomføringen av

- Stortingsvalg
- Fylkestingsvalg
- Kommunestyrevalg
- Direkte valg til kommunedelsutvalg
- Sametingsvalg
- Lokalvalg til Longyearbyen

Systemporteføljen brukes også av Valgdirektoratet til oppfølging av valggennomføring, samt til rapportering til 3. parter.

EVA - Bruksområder

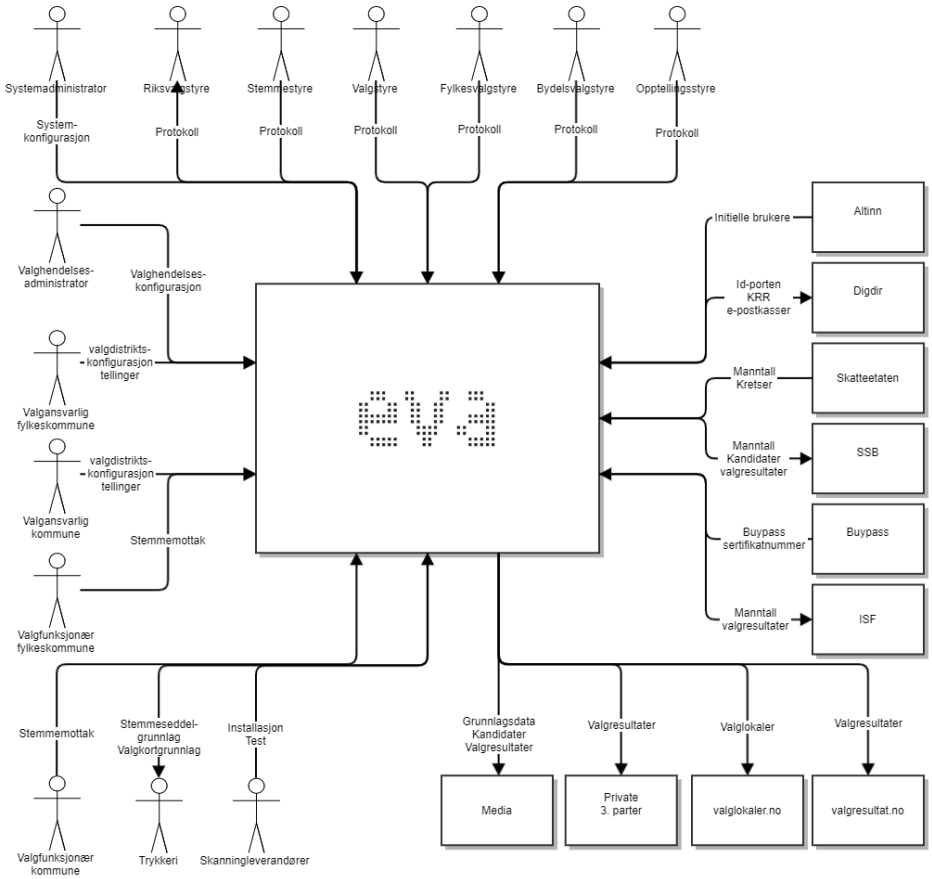
Valgavviklingen foregår over 4 faser og EVA porteføljen støtter alle disse fasene:

- Forberedelsesfasen
  - EVA Admin
- Stemmegivningsfasen
  - EVA Admin
- Opptellingsfasen
  - EVA Admin
  - EVA Skanning
  - EVA Resultat
- Valgoppgjørsfasen
  - EVA Admin
  - EVA Resultat

Som det framgår av oversikten over brukes EVA Admin i alle faser, mens andre deler av porteføljen kun brukes for utvalgte faser.

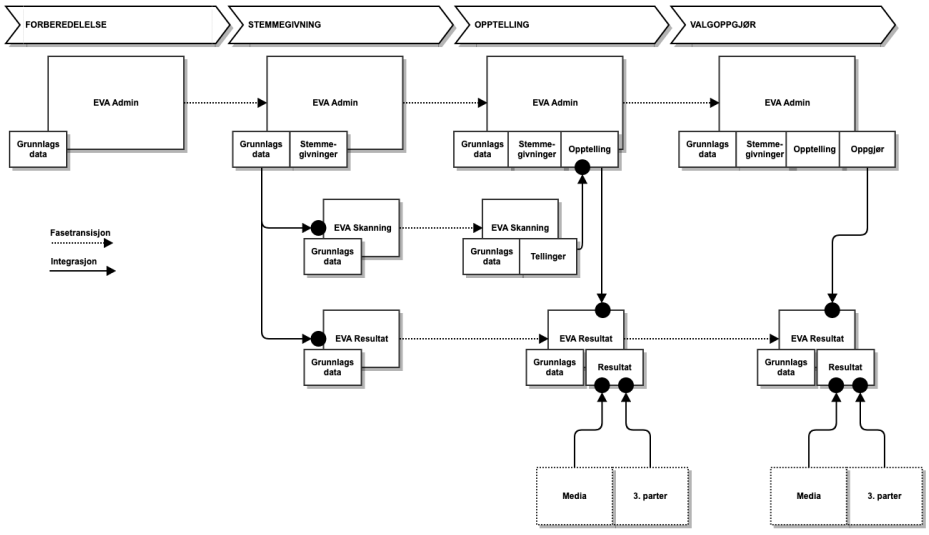
Kontekstdiagram

Diagrammet under beskriver de aktører og systemer / etater som har interaksjon og/eller interesser i/for EVA porteføljen.



Verdikjedekontekst

Diagrammet beskriver EVA systeminteraksjon i valggjennomføringsverdikjeden, kun hovedmuduler og aktører er inkludert i diagrammet.



Dette dokumentet omhandler kun EVA Skanning, de andre modulene beskrives i separate dokumenter spesifikke for den enkelte modul.

Overordnet beskrivelse - EVA Skanning

EVA Skanning

EVA Skanning er IT-systemet som brukes for å tilby systemstøtte for maskinell lesing og gjenkjenning av stemmesedler ved opptelling av stemmesedler.

EVA Skanning benyttes av kommuner og fylkeskommuner som ønsker å lese stemmesedlene maskinelt, fremfor å telle manuelt. Kommunene og fylkeskommunene kan selv vurdere om det er hensiktsmessig for de å benytte seg av skanningløsningen. Vurderingene som gjøres tar utgangspunkt i hensyn som risiko, kostnader og effektivitet. Cirka halvparten av kommunene og alle fylkeskommunene benytter EVA Skanning. Det er gjerne mindre kommuner som velger å ikke benytte skanningløsningen, da antallet stemmesedler som skal telles opp er for få for å veie opp for kostnadene ved en installasjon. Disse kommunene kan benytte EVA Admin for en manuell registrering av opptellingene.

EVA Skanning installeres på lokale maskiner ute i kommune. Kommunene kan velge å benytte tredjepartsaktører, som er kvalifisert av Valgdirektoratet gjennom rammeavtale, til kjøp av utstyr, installasjon og/eller bistand.

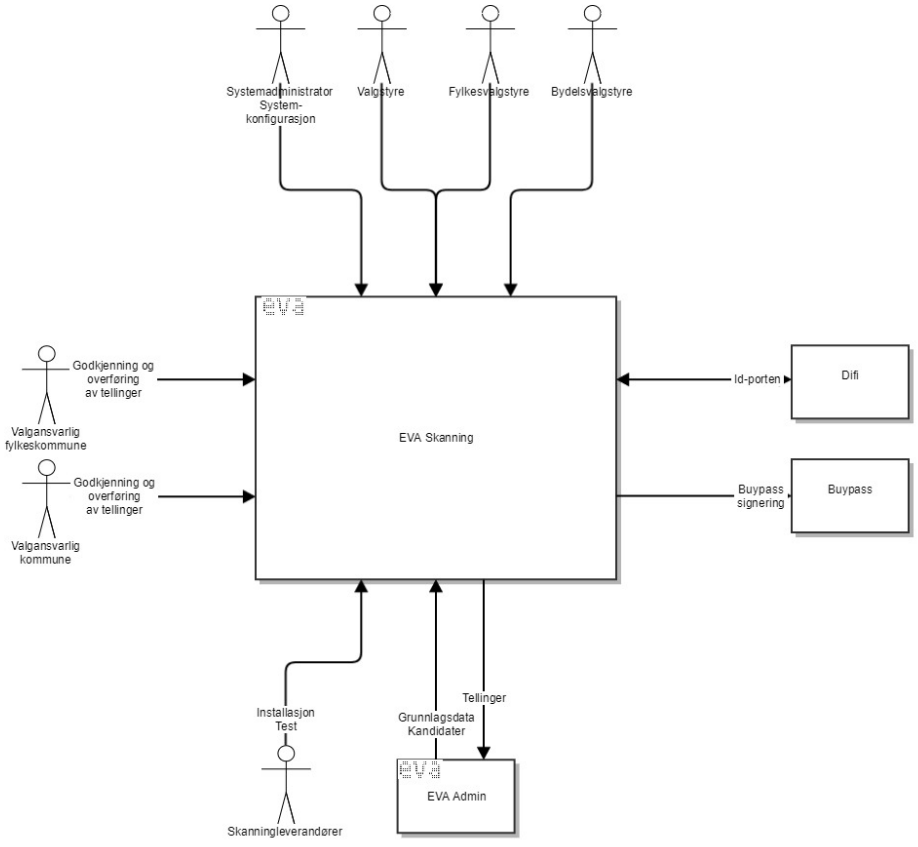
Bruksområder

EVA Skanning brukes i opptellingsfasen av valgavviklingen til maskinell lesing og gjenkjenning av stemmesedler. Applikasjonen dekker 4 hovedfunksjoner for dette området og er tilsvarende delt inn i 4 moduler:

- Jobbstyring
  - Styring og oversikt over hvilke bunker / kasser som skannes på gitte tidspunkt
  - Ferdigstilling av telling og overføring av telleresultater til EVA Admin
- Skanning
  - Skanning og tolkning av stemmesedler for et gitt valg der:
    - Parti identifiserers
    - Endringer på stemmesedler identifiseres
    - Stemmesedler akkumuleres opp i et telleresultat
- Verififisering
  - Funksjonalitet for menneskelig verifikasjon og tolkning av stemmesedler som ikke er maskinelt lesbare, eller der resultatet av den maskinelle lesingen er usikkert
- Stikkprøve
  - Funksjonalitet for å manuelt kontrollere at EVA Skannings tolkning av stemmesedler er likelydende med faktiske stemmesedler

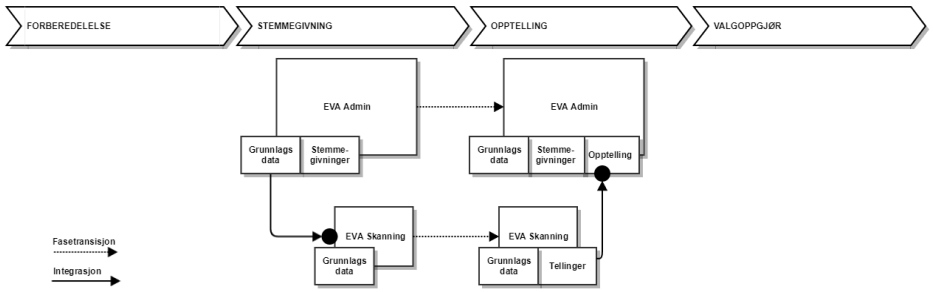
Kontekstdiagram

Kontekstdiagrammet beskriver de aktører og systemer som bruker EVA Skanning



Verdikjedekontekts

Diagrammet beskriver EVA systeminteraksjon i valggjennomføringsverdikjeden med fokus på EVA Skanning



Sikkerhetsstrategi

Kontekst for sikkerhetsstrategi

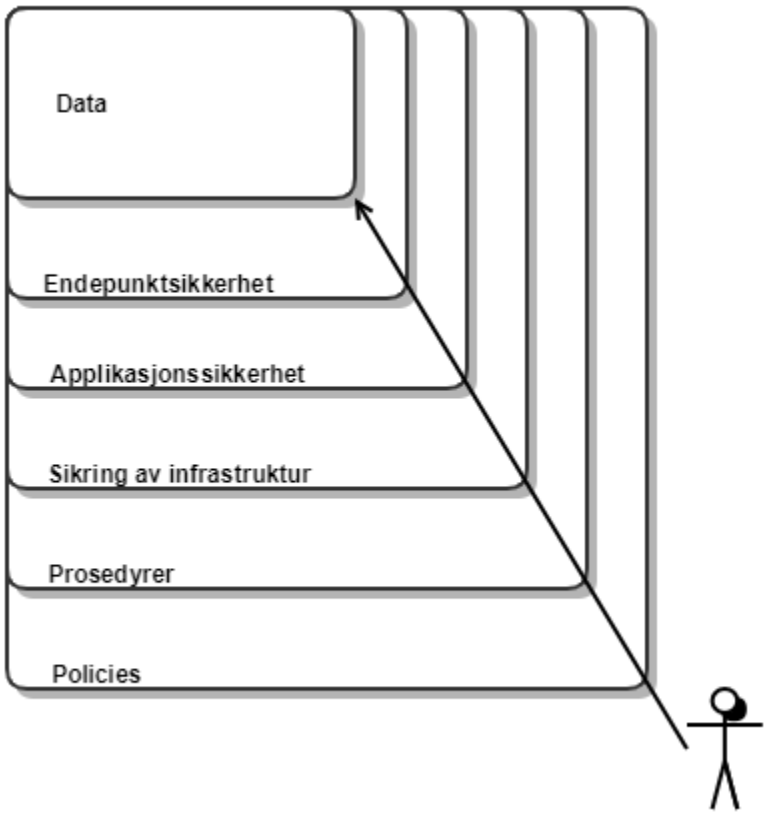
Sikkerhetsstrategien for EVA porteføljen inngår som en del av Valgdirektoratets policies for informasjonssikkerhet, sikkerhet og styring og kontroll.

Sikkerhet gjennom lagdeling

Applikasjonene sikres gjennom lagdeling som konseptuelt kan deles inn som følger:

- Policies
- Prosedyrer
- Sikring av infrastruktur
- Applikasjonssikkerhet
- Endepunktsikkerhet

Skisse som illustrerer lagdelingen som er beskrevet over



Policies og prosedyrer

Valgdirektoratet har utarbeidet policies og prosedyrer for sikkerhet.

Policy

En policy beskriver overordnede føringer og prinsipper for etablering, oppfølging og forbedring av et funksjonsområde.

Prosedyre

En prosedyre beskriver konkret framgangsmåter for å etterkomme en eller flere policies.

Valgdirektoratets policies

Valgdirektoratet har utarbeidet følgende policies med tilhørende prosedyrer:

- "Policy for informasjonssikkerhet"
- "Policy for sikkerhet"
- "Policy for styring og kontroll"

i tillegg til et sett med prosedyrer og retningslinjer, samt implementasjoner som oppfyller prinsipper og retningslinjer gitt i policies.

Lokale prosedyrer

Det er kommuner og fylkeskommuner som selv drifter EVA Skanning-løsningen. Valgdirektoratet har veiledere som utgangspunkt for lokale prosedyrer:

- Forberedelser til bruk av EVA Skanning
- Sikkerhetsveileder
- Installasjonsguide

Kommunikasjon over internett

Kryptering av kommunikasjon over internett

Samtlige av Valgdirektoratets EVA produkter kommuniserer over internett, enten som avsender av informasjon eller som mottaker.

EVA produktene kommuniserer utelukkende på HTTPS-protokollen (Hypertekst Transfer Protocol Secure) der kommunikasjonen skjer over åpne nettverk (internett).

For en nærmere beskrivelse av HTTPS, se Store norske leksikon sin beskrivelse av protokollen, det gis ingen utdypende forklaring av protokollen i systemdokumentasjonen - annet enn at protokollen brukes.

Sertifikater / PKI

Valgdirektoratet har anskaffet nødvendige sertifikater fra *anerkjente sertifikatutstedere* (CAs) for å understøtte sikker kommunikasjon over internett og ivareta behovet for:

- Autentisering
- Kryptering (asymmetrisk)

For en nærmere beskrivelse av sertifikater/PKI se Store norske leksikon sin beskrivelse av PKI.

Klientsertifikater

Valgdirektoratet utsteder klientsertifikater som alle som skal kommunisere med EVA produktene må installere, dette er en ekstra sikkerhetsmekanisme for å begrense tilgangen til EVA produktene samt tilleggs-autentisere alle brukere av systemene. Dette påvirker ikke mekanismene beskrevet over.

Funksjonelle moduler

Funksjonelle moduler beskriver modulene som brukerne interagerer med og som manifisterer seg som skjermbilder, EVA Skanning er delt inn i 4 moduler:

- Jobbstyring
- Skanning
- Verifisering
- Stikkprøve

I praksis er disse modulene egentlig små applikasjoner, men i kontekten EVA Skanning som èn applikasjon omtales de som moduler.

Utover beskrivelsene som er gitt under henvises det til http://valgmedarbeiderportale.valg.no, brukerveiledning EVA for nærmere beskrivelser av funksjonalitet i EVA Skanning.

Jobbstyringsmodul

Formål

EVA Jobbstyring er knyttet sammen med valgadministrasjonssystemet EVA Admin, og benyttes til å definere opp hva som skal telles når, holde oversikt underveis i opptellingsprosessen og overføre resultater til EVA Admin.

Følgende hovedfunksjoner fins i EVA Jobbstyring

- Starte og fullføre tellinger
- Avklare eventuelle duplikater
- Overvåke fremdriften i skanning og verifisering
- Overføre telleresultater til EVA Admin

Modulbeskrivelse

I det følgende beskrives de viktigste konseptene i jobbstyringsmodulen.

Start en telling

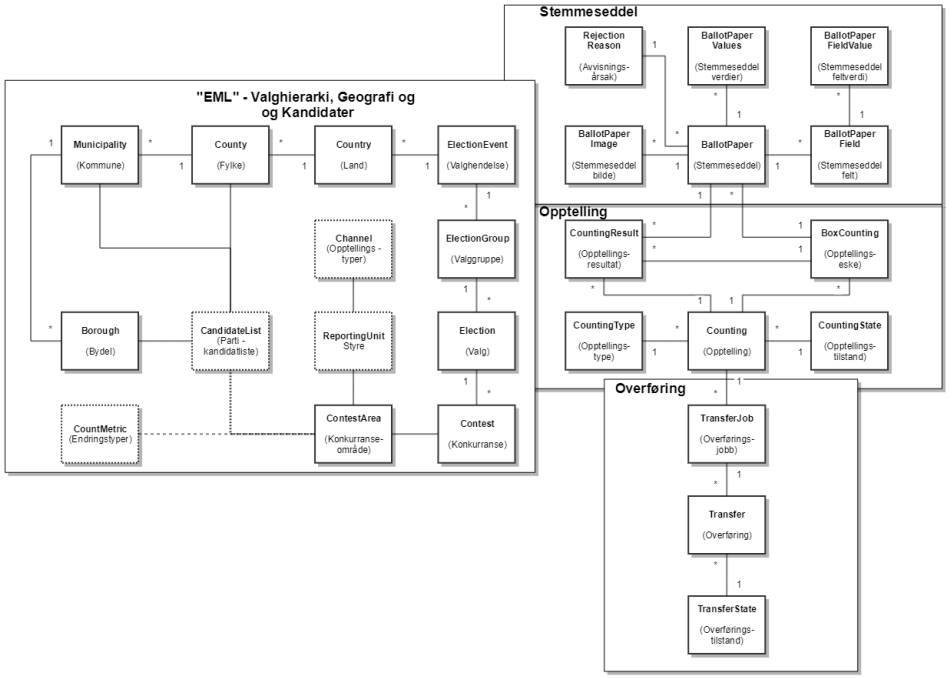
Jobbstyring er modulen som starter en telling og avslutter en telling. En telling opprettes for:

- Et gitt valg
- Et gitt valgdistrikt (kommune eller fylkeskommune)
- En gitt krets
- En gitt opptellingskategori

Informasjonen om tilgjengelige muligheter for typene over er gitt ved konfigurasjonen av EVA Skanning.

Når tellingen er instansiert aventer jobbstyringsmodulen at telling påbegynnes og avsluttes av skanningmodulen.

I domenemodellen under er det tabellene Counting i Opptelling med en gitt CountingType som opprettes, typer hentes fra domenet "EML" - Valghieraki, Geografi og kandidater



Avslutt og overfør en telling

Jobbstyring lytter etter tellinger som er ferdige dvs. tellinger der alle kasser for en gitt telling er gjort. Når en ferdig telling oppdages kan den ha to tilstander:

- Tellingen er ferdig og alle stemmesedler er gjenkjent
- Tellingen er ferdig, men enkelte stemmesedler er ikke entydig tolket og krever manuell verifisering

I førstnevnet tilfelle kan tellingen godkjennes og overføres til EVA Admin, i sistnevnte tilfelle brukes verifiseringsmodulen for å manuelt verifisiere og ta stilling til stemmesedler som ikke er entydig tolket.

Når en telling er ferdigstilt og ingen stemmesedler ligger til verifisering kan telleresultatet overføres til EVA Admin. Overføring er kun tilgjengelig for rollen valgansvarlig. Før tellefilen overføres til EVA Admin signeres den av valgansvarlig med dennes personlige Buypass smartkort.

I domenemodellen over vil jobbstyring opprette tabeller i Overføring.

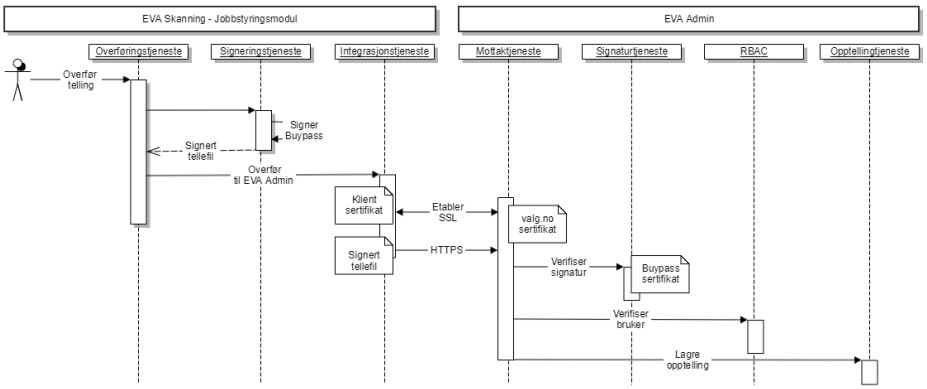
Oversending av telleresultater til EVA Admin

Ved ferdigstilling av en gitt telling skal telleresultatet oversendes til EVA Admin.

Overføringen av opptellingsfilen sikres på flere måter:

- Ved roller: Det er kun valgansvarlig som kan overføre opptellingsfiler fra EVA Skanning til EVA Admin
- Ved signering av telleresultat: Valgansvarlig må signere tellefilen ved hjelp av Buypass signeringstjeneste med Buypasskort
- Ved tilgangskontroll: EVA Skanning må inneha et klientsertifikat for å få tillatelse til å kommunisere med EVA Admin
- Ved kryptering: Tellefilen oversendes over HTTPS med asymmetrisk kryptering
- Ved verifisering i EVA Admin: Ved mottak av tellefil vil EVA Admin verifisere at avsenders signatur har opphav i forventet Buypass-sertifikat
- Ved autentisering: EVA Admin: Ved mottak av tellefil vil EVA Admin kontrollere om signatur fra avsender stemmer overens med bruker som er definert som valgansvarlig i EVA Admin

Skissen illustrer en forenklet framstilling av oversending av opptellingsresultat fra EVA Skanning (jobbstyringsmodulen) til EVA Admin med fokus på sikkerhetsmekanismer (skissen inneholder begge systemer for å sikre helheten)



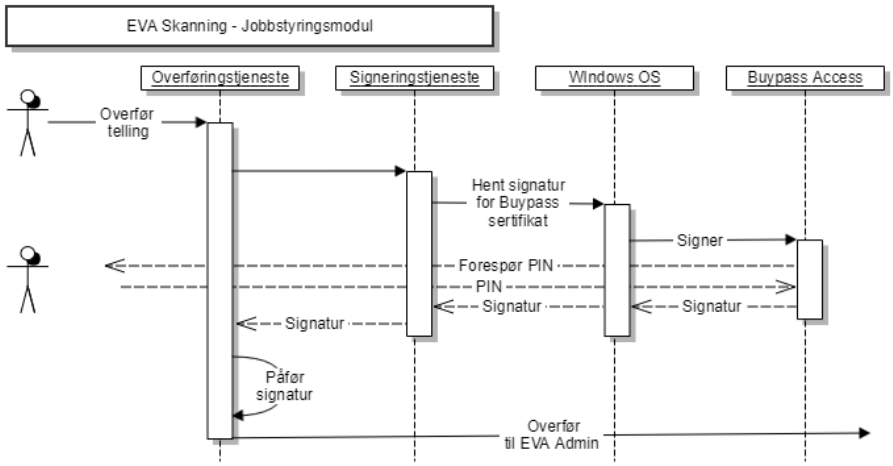
Signering av opptellingsresultat

Før en opptelling kan overføres fra EVA Skanning til EVA Admin må opptellingsfilen signeres av valgansvarlig ved hjelp av et Buypass smartkort.

EVA Skanning er integrert med Buypass Access Enterprice som besørger signering av opptellingen ved å bruke Buypass smartkort sertifikatet. Buypass Access Enterprice inkluderes i EVA Skanning installasjonspakke og installeres sammen med EVA Skanning på klienten. Beskrivelse av Buypass Access Enterprice finnes her: https://buypassdev.atlassian.net/wiki/spaces/BAS/pages/9666985/Buypass+Access+Manager+-+Systembeskrivelse

Merk at det kun er tjenesten for *signering* som benyttes, Buypass brukes *ikke til kryptering*.

Skisse som gir en forenklet illustrasjon av signeringsforespørsel mot Buypass Access Enterprice



Buypasskort

Buypasskort er personlige smartkort og er knyttet til en persons identitet og skal ikke deles eller overdras.

Buypass dokumentasjon: https://www.buypass.no/produkter/buypass-id-smartkort

Buypasskortleser

Man må bruke en smartkortleser for å lese Buypasskortet, denne kan være kjøpt av Buypass eller andre leverandører, evt. integrert i klient.

Skanningmodul

Formål

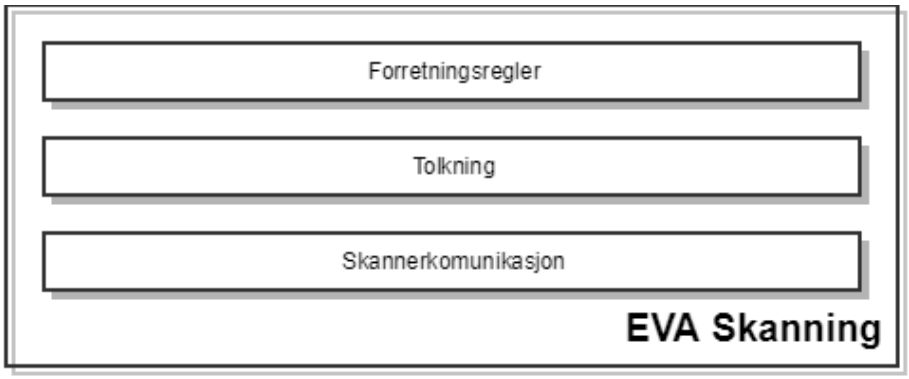
EVA Skann benyttes når stemmesedlene skal leses inn med skanneren og fortolkes maskinelt.

Modulbeskrivelse

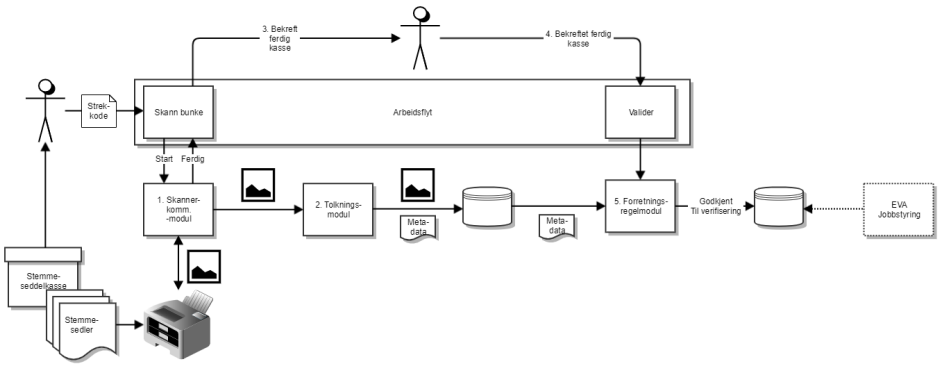
Skanningmodulen:

- Starter skanner og mottar tilbakemdelding når dokumentskanner har skannet en bunke i en kasse
- Mottar bilder fra dokumentskanner og utleder informasjon fra bildene
- Anvender forretningsregler på informasjonen utledet fra bildene og vurderer korrekthet og behov for manuell verifikasjon
- Signaliserer til EVA Jobbstyring når en kasse er ferdig skannet (se Konfigurasjon av opptellingslogistikk)

Konseptuelt kan skanningmodulen deles inn i 3 funksjonsområder som utgjør maskinell lesing av stemmesedler:



EVA Skanning orkestrerer arbeidsflyten ved maskinell lesing av stemmesedler fra papirstemmeseddel til maskinelt tolket stemmeseddel representert ved bilde og metadata:



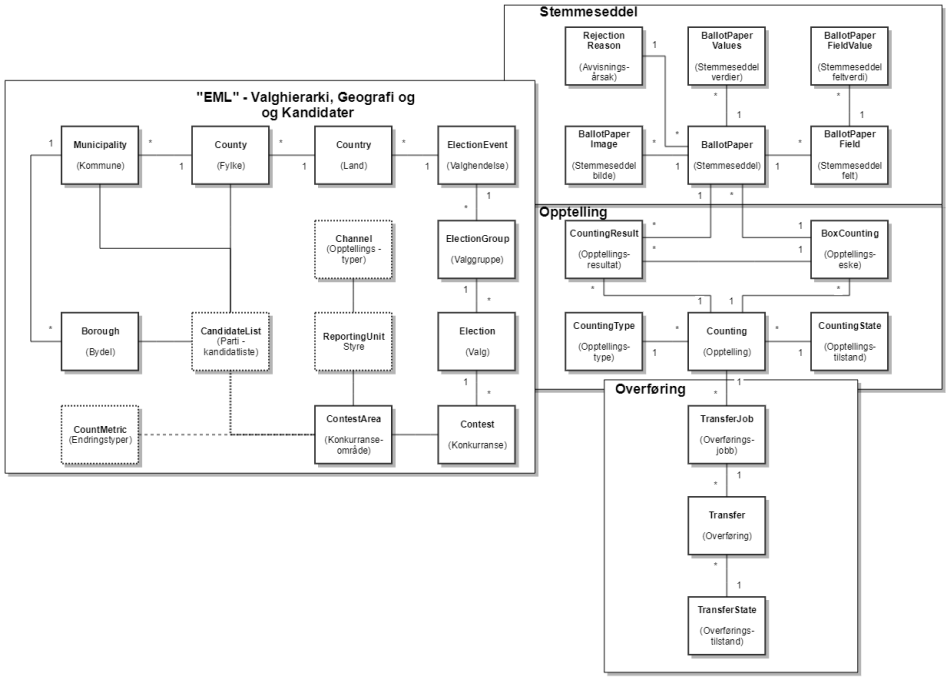
Forutsetning:

En telling er startet i EVA Jobbstyring, se Konfigurasjon av opptellingslogistikk.

Arbeidsflyt:

- En skanningoperatør skanner en strekkode for en kasse med stemmesedler
- EVA Skanning starter kommunikasjon med skanner (1) og skanneren begynner å lese / fotografere stemmesedler
- Bilder mottas fra dokumentskanner og videreformidles til tolkningsmodulen (2), der:
  - Bilde vurderes opp mot forventet resultat (mal)
  - Felter som er definert som maskinlesbare tolkes og metadata om feltene utledes
- Bilde og metadata lagres i databasen
- Når alle stemmesedler som er matet inn i skanneren er ferdig behandlet bes skanningoperatør om å bekrefte at kassen er ferdig talt eller ikke (3)
  - Dersom skanningoperatør avkrefter at kassen er ferdig mates skanneren med en ny bunke med stemmesedler og prosessen over gjentar seg
- Dersom skanningoperatør bekrefter at kassen er ferdig, vurderes metadata mot forretningsregler og valgkonfigurasjon (EML) (5) for å avgjøre om resultatet er:
  - I tråd med forretningsreglene for valget stemmesedlene gjelder for
  - I tråd med konfigurasjonen av valget stemmesedlene gjelder for (EML)
- Stemmeseddelen merkes som godkjent eller ikke godkjent, i hvilket tilfelle manuell verifisering må gjennomføres

I domenemodellen under er det området Stemmeseddel som befolkes med informasjon om skannede stemmesedler, samt opptellingsesker og opptellingsresultet. Informasjon fra "EML" - Valghierarki, Geografi og Kandidater brukes for i vurderingen

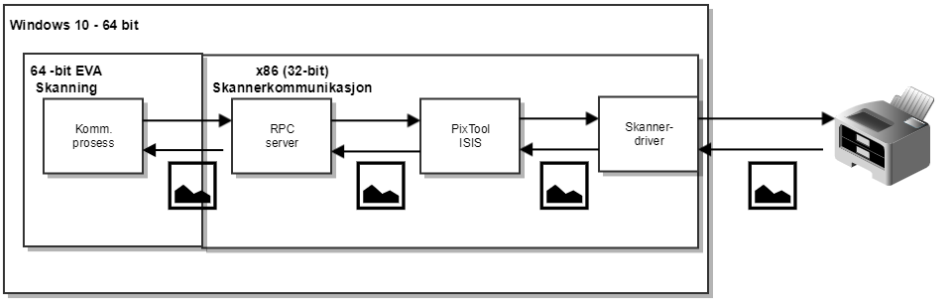


Skannerkommunikasjonsmodul

EVA Skanning bruker GRPC "Remote Procedure Call" for å kommunisere med dokumentskanner ved hjelp av ISIS-grensesnittet (Image and Scanner Interface Specificaton).

Løsningen for skannerkommunikasjon er valgt for løse 64 / 32 bits problemstillinger mellom EVA Skanning (64 bits) og skannerdrivere (32 bit).

Skissen illustrerer kommunikasjon mellom EVA Skanning og dokumentskanner



Eva Skanning instruerer kommunikasjonsmodulen om at utføre skanning. Kommunikasjonsmodulen instansierer en 32 bits GRPC server / prosess som kommuniserer videre ved hjelp av Pixtool/ISIS med skannerdriver og dokumentskanner.

Instruksjoner og bilder formidles gjennom disse leddene.

Kommunikasjonen mellom EVA Skanning kommunikasjonsmodul og GRPC-server er beskyttet med SSL-sertifikat for å autentisere klient (EVA Skanning) og kryptere trafikk.

Tolkningmodul

EVA Skanning bruker en tolkningsmodul for å tolke stemmeseddelbilder, modulen er satt sammen av et utvalg open source-rammeverk:

- OpenCv (Open Source Computer Vision Library) for bildeprosessering
- Tesseract OCR-engine for OCR lesing av stemmesedler
- Tensorflow or Keras for trening og modellering av håndskriftsgjenkjenning
- ONNX for kjøring av modeller for håndskriftsgjenkjenning

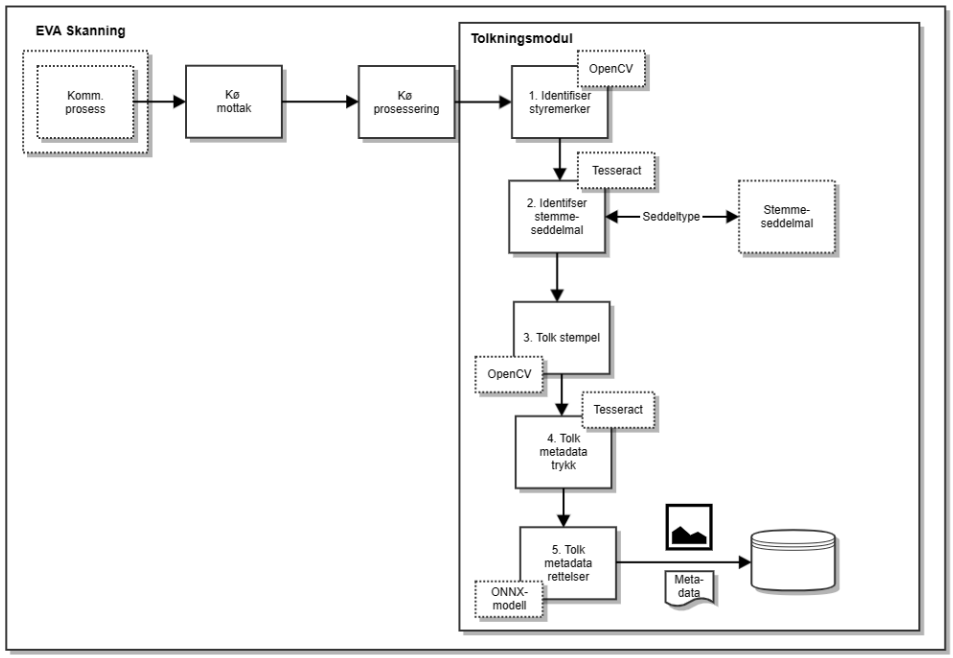
Prosessen bruker to hovedartifakter for å tolke stemmeseddel:

- Stemmeseddelbilde av forside og bakside mottatt fra dokumentskanner
- Stemmeseddelmal som inneholder hvordan gyldige stemmesedler forventes å se ut, samt hvor på stemmeseddelen metadata skal utledes

Basert på stemmeseddelbilde og stemmeseddelmal produseres et tredje artifakt

- Metadata som er resultatet av tolkningen av stemmeseddelbilde

Skissen gir en forenklet illustrasjon av prosess og biblioteker brukt for å tolke stemmeseddelbilde, metadata utledes i de forskjellige stegene



Prosessbeskrivelse:

Transaksjoner (stemmeseddelbilder) fra kommunikasjonsmodulen legges på en mottakskø og flyttes umiddelbart til en prosesseringskø der tolkningsmodulen henter stemmeddelbilder

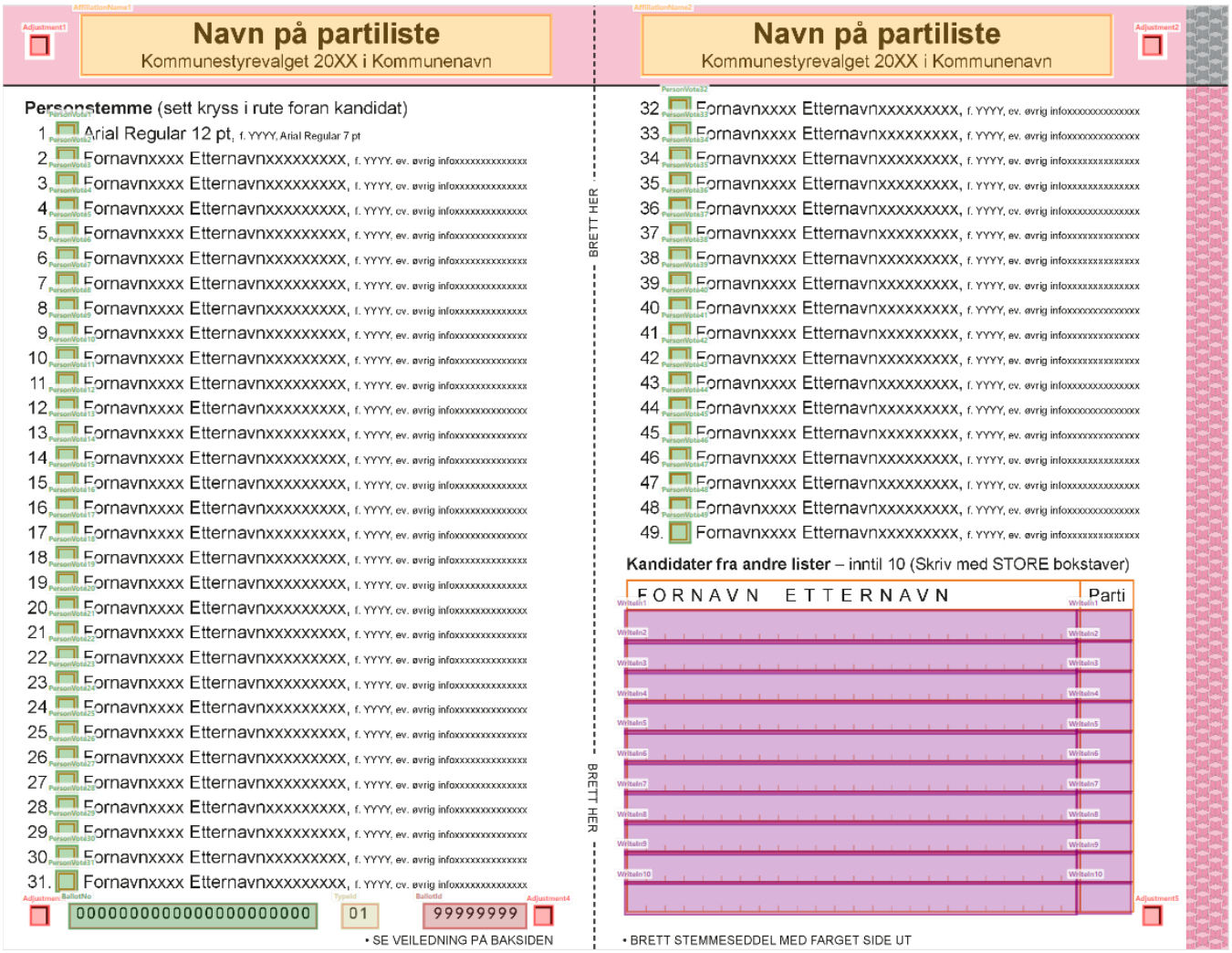
1. Styremerker på stemmeseddelen identifiseres for at tolkningsmodulen skal kunne orientere stemmeseddelen
2. Stemmeseddeltype identifiseres og mal med samme stemmeseddeltype identifiseres i mal-biblioteket. Malen angir hvilke felter og områder tolkningsmodulen skal lese og tolke på stemmeseddelen
3. Fyllingsgrad for stempel tolkes, fyllingsgrad brukes all den tid kommuner og fylkeskommuner bruker forskjellige stempel
4. Metadata utledes fra de feltene som skal tolkes på stemmeseddene
5. Rettelser på stemmeseddelen utledes og tolkes ved hjelp av ONNX og Valgdirektoratets modell for håndskrift. (Modellen er manuelt definert av Valgdirektoratet for å sikre kontroll med bokstavtolkning)
6. Metadata kobles til stemmeseddelbilder og lagres i databasen, status vil være enten
    a. Godkjent - ingen videre manuell prosesering er påkreved
    b. Krever manuell verifisering - i de tilfeller der en eller flere av tolkningsforsøkene mislykkes

Maler

For hver type stemmeseddel (f.eks todelt partistemmeseddel kommunestyrevalg) defineres en mal som forteller tolkningsmotoren hvor den skal utlede informasjon fra.

Disse malene defineres av Valgdirektoratet og distribueres sammen med EVA Skanning i installasjonsoppsettet.

Illustrasjon av mal for todelt partistemmeseddel kommunestyrevalg, felter som tolkes er markert med farge:



Eksempel på mal-xml som distribueres med EVA Skanning installasjonspakke (eksempelet er ikke uttømmende men illustrerer hovedelementene i malen:

```
<?xml version="1.0" encoding="utf-8"?>
<template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
id="Kommune2" electionId="02" friendlyName="Kommune 2-delt" version="2023.0" DPI="300" typeId="01"
physicalWidthMm="288" physicalHeightMm="223" scanThreshold="120" markerTemplate="false" xmlns="urn:scan-omr:
template">
  <inside>
    <adjustmentFields type="UpsideDownL" requiredNumber="0" strokeWidth="0">
      <adjustmentField id="Adjustment1" topLeft="79, 84" bottomRight="126, 134" />
      <adjustmentField id="Adjustment2" topLeft="3191, 84" bottomRight="3238, 134" />
      ...
    </adjustmentFields>
    <writeInFields isContourVisible="false">
      <writeInField id="WriteIn1">
        <affiliationField topLeft="3004, 1690" bottomRight="3152, 1774" />
        <nameField topLeft="1741, 1690" bottomRight="3001, 1778.4054" />
      </writeInField>
      ...
    </writeInFields>
    <textField id="AffiliationName1" topLeft="220.71504, 26.678003" bottomRight="1530.7004, 190.68678" />
    <textField id="AffiliationName2" topLeft="1788.5522, 26.473679" bottomRight="3097.6743, 191.0074" />
    <textField id="BallotId" topLeft="1178, 2509" bottomRight="1461, 2574" />
    <textField id="TypeId" topLeft="949, 2509" bottomRight="1047, 2574" isRecognitionField="true" />
    <textField id="BallotNo" topLeft="187, 2509" bottomRight="875, 2574" />
    <personVoteFields>
      <personVoteField id="PersonVote1" topLeft="152, 327" bottomRight="207, 382" candidateNumber="1" />
      <personVoteField id="PersonVote2" topLeft="152, 397" bottomRight="207, 451" candidateNumber="2" />
      ...
...
```

Forretningsregelmodul

Forretningsregelmodulen validerer om stemmesedler og endringer på stemmesedler er gyldige for en gitt stemmeseddeltype for et gitt valg.

Resultatet av valideringen har to utfall:

1. Stemmeseddel og endringer på stemmeseddel er entydige og korrekte i forhold til regelsett for den gitte stemmeseddelen for det gitte valget - disse stemmesedlene er ferdig behandlet
2. Stemmeddel og/eller rettelser på stemmeseddel er ikke entydige, eller ikke korrekte i forhold til regelsett for den gitte setemmeseddelen for det gitte valget - disse stemmesedlene krever manuell verifisering i verifiseringsmodulen - EVA verifiser

Forretningsregler

De fleste forretningsreglene kontrollerer mottatte data mot *valgkonfigurasjonen* som er en del av Konfigurasjon av applikasjon. Opphavet til valgkonfigurasjonen er EVA Admin.

Overordnet kreves det at attributene på stemmeseddelen stemmer overens med valgkonfigurasjonen, og at rettelser er gjort i henhold til lov og forskrift som regulerer hvilke endringer som kan gjøres på hvilke typer stemmesedler for en gitt valgtype.

Forretningsreglene anvendes automatisert èn gang i EVA Skanning, der disse ikke gir et entydig resultat må det gjøres en manuell vurdering og/eller verifisering og/eller endring i verifiseringsmodulen - EVA Verifiser.

Universelle forretningsregler

Stempel

Stemmeseddelen skal være stemplet

Stemmeseddelnummer

Stemmeseddelnummer er delt inn i flere elementer:

- Valghendelsesid: 6 siffer - må være lik som valghendelsesid i valgkonfigurasjon (EML)
- Valggruppe: 2 siffer - må være lik som valggruppe i valgkonfigurasjon (EML)
- Valg: 2 siffer - må være lik som valg i valgkonfigurasjon (EML)
- Valgdistriktid: 6 siffer - må være lik som valgdistrikt i valgkonfigurasjon (EML)
- Partinummer: 4 siffer - må være lik som partinummer i valgkonfigurasjon (EML)
- kontrollsiffer: 2 siffer - checksum

StemmeseddelId

Stemmeseddelid angir type stemmeseddel samt antall falser / deler stemmeseddelen er delt opp i

Løpenummer

Løpenummer skal være unikt for en gitt stemmeseddel

Partinavn

Partinavn leses fra topplinjen og må stemme med partinavn som utledes fra stemmeseddelnummer (partinummer), og må finnes i EML for valgdistriktet for det valget som skannes.

Unntak: All den tid avkrysningsstemmesedler ikke er partispesifikke gjøres denne kontrollen ikke for avkrysningsstemmesedler

Valg

Valg leses fra topplinjen (f.eks. Kommunestyrevalg 2023 i Bodø), og må stemme med valg som utledes fra stemmeseddelnummer, og må være korrekt for det valget som skannes.

Valgspesifikke forretningsregler

Valgspesifikke forretningsregler er regler som beskriver hvilke gyldige *rettelser* stemmegiver kan gjøre på en stemmeseddel, også disse forretningsreglene innebærer kontroll mot valgkonfigurasjon.

Hvilke valgspesifikke rettelser velger skal tillates å gjøre defineres i valgkonfigurasjonen (EML) og fasiliteres ved stemmeseddelens grafiske uttrykk og tilhørende mal.

For eksempel vil en angivelse av en kadidat i kandidater fra andre lister ("slengere") kontrolleres mot gyldige kandidater i valgkonfigurasjonen (EML) - dersom valgtypen tillater denne typen endring.

Verifiseringsmodul

Formål

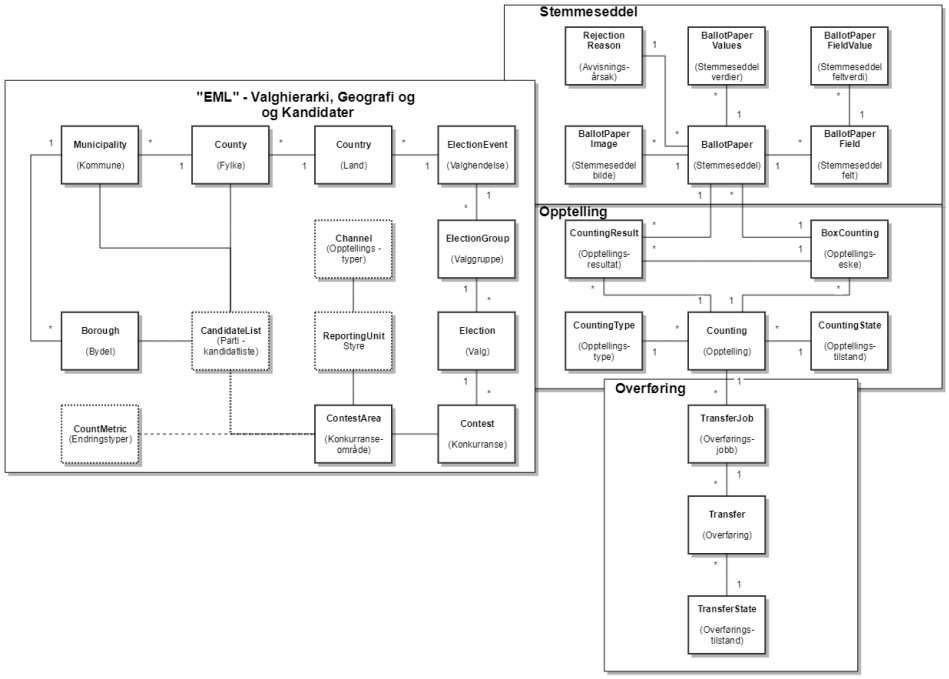
EVA Verifiser er et verktøy dedikert til effektiv verifikasjon og endring av tvetydige eller uforståelige stemmesedler ut i fra menneskelig skjønn i et forsøk på å tolke velgerens intensjon. Verifiseringsprosessen er tilpasset det valget som er under opptelling og reglene for behandling er tilpasset det geografiske nivået som utgjør tellingen.

Modulbeskrivelse

Verifiseringsmodulen bistår brukeren i tolkning av stemmesedler som ikke har latt seg entydig tolke maskinelt. Modulen vil der det er mulig presentere alternativer, men det er verifiseringsoperatøren som tar alle avgjørelser om hvorvidt en stemmeseddel skal forkastes eller godkjennes med endringer.

Verifiseringsmodulen bruker en kombinasjon de skannede stemmeseddelbildene, verdiene som er tolket fra dette, samt konfigurasjonen av EVA Skanning for å presentere brukeren for hvilke vurderinger skanningmodulen har gjort, hvor skaningmodulen ikke oppnår entydig resultat, samt verdiforslag der dette er mulig basert på konfigurasjonen av EVA Skanning. Verifiseringsmodulen vil ikke overprøve den menneskelige tolkningen og eventuelle endringen.

I domenemodellen under er det den lagrede informasjonen fra området Stemmeseddel, samt "EML" - Valghierarki, Geografi og Kandidater som brukes for å presentere verifiseringsgrunnlag for brukeren.



Stikkprøvemodul

Formål

Stikkprøvemodulen brukes for å manuelt verifisere at EVA Skannings tolkinng av en stemmeseddel er likelydende med den faktiske stemmeseddelen.

Beskrivelse

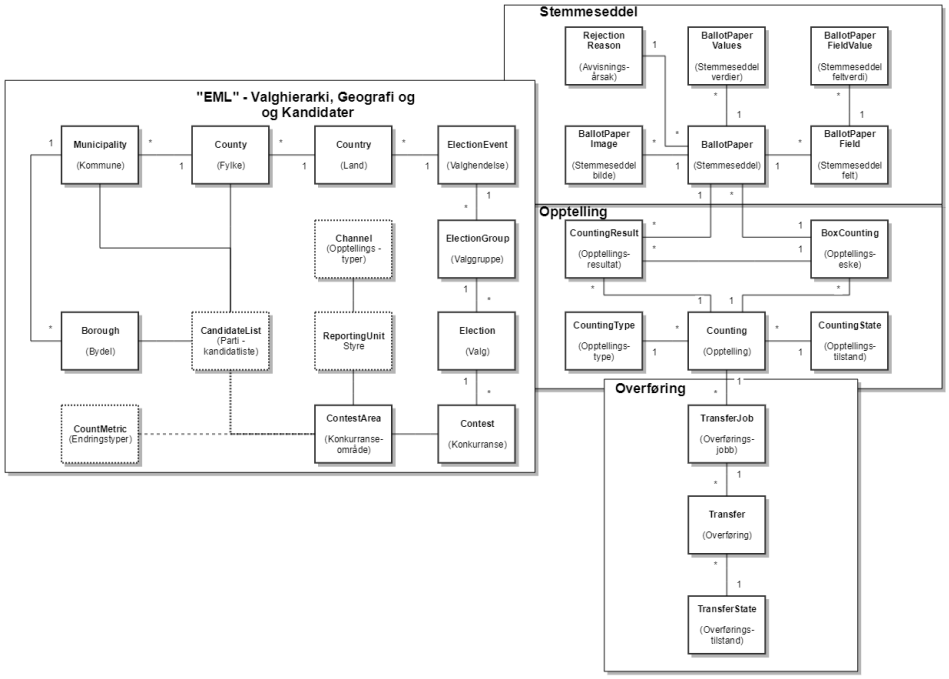
Stikkprølvemodulen bruker strekkodelapp som input for å hente opp stemmesedler i en gitt kasse (se Konfigurasjon av opptellingslogistikk).

Bruker kan deretter bla gjennom hver enkelt stemmeseddel digitalt og fysisk for å:

- se hvorvidt stemmeseddelen kun har vært gjenstand for maskinell behandling, eller om stemmeseddelen har vært gjenstand for manuell verifisering
- Sammenlikne stemmeseddelbilde med fysisk seddel
- Sammelikne EVA Skannings tolkning av stemmeseddel opp mot stemmeseddelbilde og fysisk seddel
- Sammenlikne kassesammendrag med eventuell fysisk telling av stemmesedler i kassen

Stikkprøvemodulen bruker en kombinasjon av de skannede stemmesedelbildene og verdiene som er tolket ut i fra disse.

I domenemodellen under er det den lagrede informasjonen fra området Stemmeseddel som bruke for å presentere informasjonen beskrevet over.

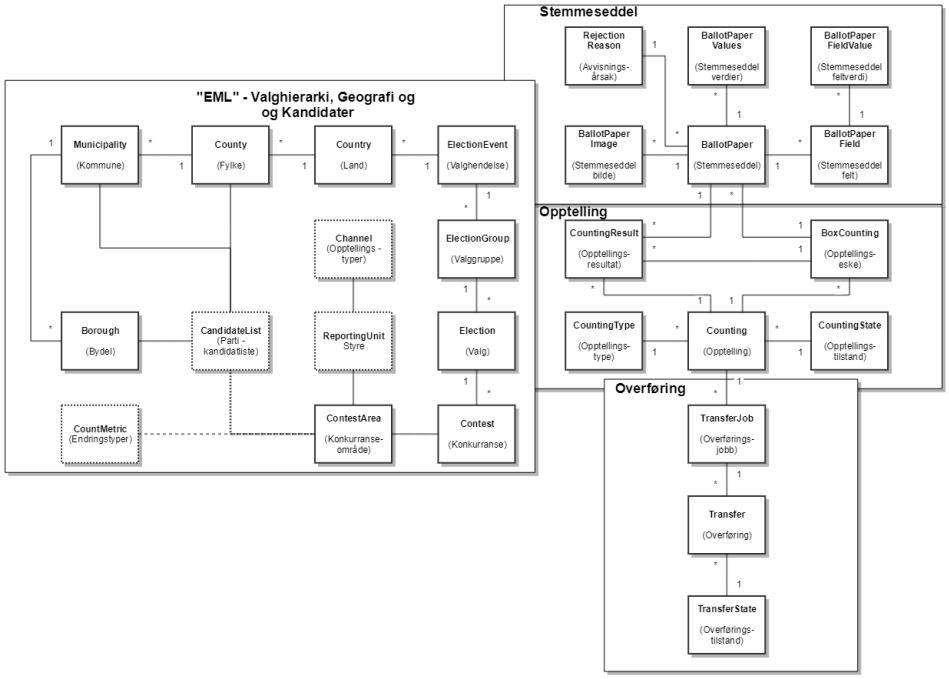


Opptellingsdomenet

EVA Skanning håndterer et begrenset domene - maskinell opptelling. Domenet er således begrenset til fire hovedområder:

- Geografi og kandidater
- Stemmeseddel
- Opptelilling
- Overføring

Det gir således mest mening å se domenet som en enkelt domenemodell, som prinsippielt er lik domenemodellen for opptelling i EVA Admin, men med tillegg av representasjon av papirstemmeseddlen og overføring av tellinger.



Arkitektur

Innledning

EVA Skanning ble opprinnelig utviklet for maskinell telling av stemmesedler i 2013 av KMD, og ble deretter kraftig revidert etter valget i 2015 opp mot valget i 2017.

EVA Skanning bygde i alle disse versjonene på bruk av applikasjonen ReadSoft Forms for OCR-lesing av stemmesedler. EVA Skanning er sterkt påvirket av bruken av ReadSoft Forms all den tid applikasjonen i stor grad er en innpakning og undertrykking av ReadSoft Forms medfølgende brukergrensesnitt til fordel for et egenutviklet brukergrensesnitt som er tilpasset funksjonsområdet for maskinell telling av stemmesedler.

Da Valgdirektoratet ble opprettet 01.01.2016 overtok direktoratet forvaltningen av produktet.

Historikken dikterer således EVA Skannings arkitektur med tildels sterke føringer fra ReadSoft Forms gjennom forretnings- og presentasjonslaget i EVA Skanning.

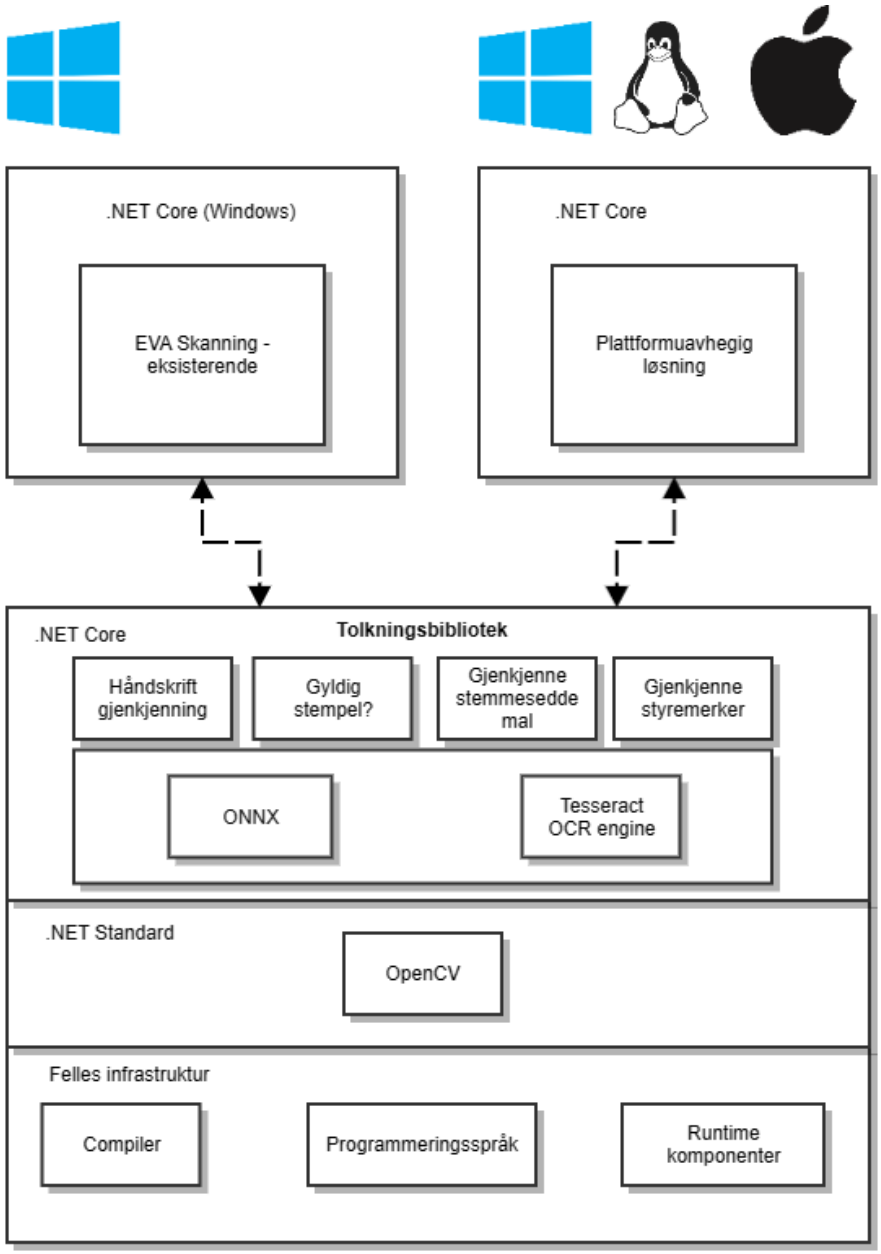
EVA Skannings utvikling

Hovedfokuset for utviklingen av EVA Skanning fra valget i 2017 opp mot valget i 2019 og videre mot senere valg er å:

- Erstatte Readsoft Forms til fordel for en sammensetning av Open Source-komponenter for å frigjøre applikasjonen fra føringene fra ReadSoft Forms
- Modularisere EVA Skanning ytteligere for å posisjonere EVA Skanning til endret infrastruktur samt muliggjøre sentralisering av tjenester som ikke krever lokal installasjon.
- Forbedre tolkning av stemmesedler, bla. a. ved å tilby skanning i farger.
- Videreutvikling av sikkerhetsfunksjonalitet rundt EVA Skanning

Arbeidet med erstaning av ReadSoft Forms ble fullført innen valget i 2019, men uten å fjerne grensesnitt definert av ReadSoft. Jobben med å fjerne bruk av resterende ReadSoft grensesnitt ble fullført innen valget 2023.

Illustrasjon av målbildet konkretisert ved introduksjon av ny tolkningsmotor som introduserer plattformuavhengighet for denne gitte komponenten:



Applikasjonskonfigurasjon

EVA Skanning konfigureres basert på et sett med konfigurasjonsfiler som er eksportert fra EVA Admin. Filene er på XML format og følger EML (Election Markup Language) standard.

Filene innheholder alle kommuner og fylkeskommuners valgkonfigurasjon som er påkrevd for å kunne gjennomføre maskinell telling av stemmesedler organisert på linje med konfigurasjonen i EVA Admin.

Normalt distribueres konfigurasjonsfilene i EVA Skaning installasjonspekken. Ved behov for ad hoc rettelser kan filen distribueres frittstående og importeres til EVA Skanning.

Overordnet inneholder settet med konfigurasjonsfiler følgende:

- Alle kommuner
- Alle kandidater
- Informasjon fra valghendelsen som har betydning for gjennomføring av maskinell telling av stemmesedler

Konfigurasjonsfilene er signert med valghendelsessertifikat fra EVA Admin, dette betyr i praksis at kun konfigurasjonsfiler som har sitt opphav i EVA Admin og som er distribuert av Valgdirektoratet kan brukes for å konfigurere EVA Skanning.

Valgkonfigurasjon

Følgende attributter er av betydning for hvordan maskinell telling av stemmesedler skal gjennomføres:

- Alle fylkeskommuner og kommuner i Norge på valggjennomføringstidspunktet (csv-fil)
- Samtlige kandidater og deres partitilhørighet for det gitte valget for det gitte valgdistriktet (kommundelsutvalg, kommune eller fylkeskommune)
- Informasjon av betydning for hvordan opptelling av stemmesedler skal gjennomføres:
    - Valghendelse (Election event)
        - Valg (Election)
            - "Konkurranse" (Contest)
                - Valgdistriktets Styrenivåer (Reporting Unit)
                    - Styrenivåets Stemmegivnings-/Opptellingskategorier (F.eks forhåndsstemmer, Valgtingsstemmer osv.) fordelt på stemmekrets der dette er relevant (ResultsReported)
            - Regler for endringer på stemmeseddelen for den gitt konkurranse

I tillegg må de faktiske tellingene konfigureres og styres, dette er beskrevet i kapittelet Konfigurasjon av opptellingslogistikk

I praksis er konfigurasjonen en instansiering av domenemodellen for opptelling i EVA Admin (se domenemodell opptelling i EVA Admin systemdokumentasjon)

Utsnitt av valgkonfigurasjon som beskrevet over fra en fiktiv valghendelse som illustrerer implementasjonen av beskrivelsen gitt over:

Valgkonfigurasjon

```
<ElectionEvent>
<EventIdentifier Id="790001">
<EventName>Kommunestyre- og fylkestingsvalget 2019</EventName>
</EventIdentifier>
<Election>
<ElectionIdentifier Id="01">
<ElectionName>Fylkestingsvalget 2019</ElectionName>
<ElectionGroup Id="01">Kommunestyre- og fylkestingsvalget 2019</ElectionGroup>
<ElectionCategory>F</ElectionCategory>
</ElectionIdentifier>
<Contest>
<ContestIdentifier Id="000007">
<ContestName>Vestfold</ContestName>
</ContestIdentifier>

/*Her angis stryenivå Fylkesvalgstyret i Vestfold, samt hvilke tellinger dette styret skal telle/kontrollere*/
<ReportingUnit>
<ReportingUnitIdentifier Id="01.01-47.07">Fylkesvalgstyret Vestfold</ReportingUnitIdentifier>
<ResultsReported>
<Status Type="FO">
<ChannelID>47.07.0701.070100.0000</ChannelID>
<Notes>Forhåndsstemmer ordinære</Notes>
</Status>
</ResultsReported>
<ResultsReported>
<Status Type="FO">
<ChannelID>47.07.0711.071100.0000</ChannelID>
<Notes>Forhåndsstemmer ordinære</Notes>
</Status>
</ResultsReported>
<ResultsReported>
<Status Type="FO">
<ChannelID>47.07.0714.071400.0000</ChannelID>
<Notes>Forhåndsstemmer ordinære</Notes>
</Status>
</ResultsReported>
<ResultsReported>
<Status Type="FO">
<ChannelID>47.07.0723.072300.0000</ChannelID>
<Notes>Forhåndsstemmer ordinære</Notes>
</Status>
</ResultsReported>
</ReportingUnit>
...

/*Her angis styrenivå Valgstyret i Horten og hvilke tellinger dette styret skal telle/kontrollere*/
<ReportingUnit>
<ReportingUnitIdentifier Id="01-47.07.0701">Valgstyret Horten</ReportingUnitIdentifier>
<ResultsReported>
<Status Type="FO">
<ChannelID>47.07.0701.070100.0000</ChannelID>
<Notes>Forhåndsstemmer ordinære</Notes>
</Status>
</ResultsReported>
<ResultsReported>
<Status Type="FS">
<ChannelID>47.07.0701.070100.0000</ChannelID>
<Notes>Sent innkomne/lagt til side</Notes>
</Status>
</ResultsReported>
<ResultsReported>
<Status Type="VF">
<ChannelID>47.07.0701.070100.0000</ChannelID>
<Notes>Fremmedstemmer</Notes>
</Status>
</ResultsReported>
...
```

Tilleggskonfigurasjon

Det er to forhold valgkonfigurasjonen (EML) ikke dekker:

- Partier på avkrysningsstemmeseddel - denne stemmeseddelen har sitt opphav i registrerte partier i partiregisteret i Brønnøysund og defineres utenfor EVA-porteføljen. Dette betyr at partier på avkrysningsstemmeseddelen må definers i en tillegskonfigurasjon i EVA Skanningapplikasjonen.
- "Opptellingsnivå" - Informasjon om hvorvidt stemmesedler telles av kommuner på vegne av fylkeskommuner (f.eks ved fylkestingsvalg) er ikke definert i valgkonfigurasjonen og må derfor defineres i en tilleggskonfigurasjon i EVA Skanningapplikasjonen.

Konfigurasjon av opptelling

Maskinell telling er en prosess med to dimmensjoner:

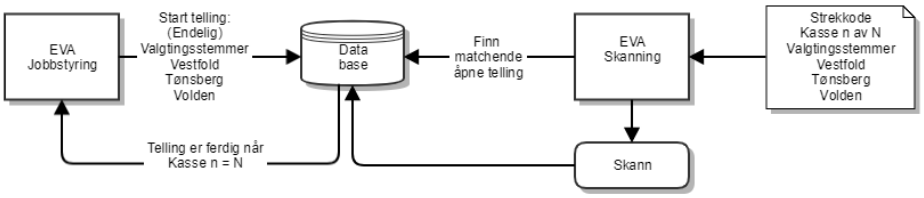
- Maskinell tolkning av stemmesedler
- Logistikk og mating av dokumentskannere med papirsedler.

For å binde disse prosessene sammen konfigureres den faktiske gjennomføringen av opptellingen gjennom strekkodelapper.

Strekkodelappene plasseres i toppen av hver eske med stemmesedler som skal telles maskinelt, disse innholder nødvendig informasjon til at EVA Skanning skal kunne knytte stemmesedler som telles til korrekt opptellingskategori og geografiske enhet.

Illustrasjon av strekkodelapp for en gitt kasse med stemmesedler av typen Forhåndsstemmer Ordinære for en fiktiv valghendelse:

Skissen illustrerer koblingen mellom EVA jobbstyring og EVA Skanning gjennom bruken av valgkonfigurasjon og strekkodelapper



Applikasjonsarkitektur

EVA Skanning applikasjonen er en distribuert .NET applikasjon som distribueres og installeres lokalt hos fylkeskommuner og kommuner som skal telle stemmesedler maskinelt.

For "Skannsenter" settes flere klienter opp mot samme database, for små installasjoner settes klient og database opp på en enkelt PC.

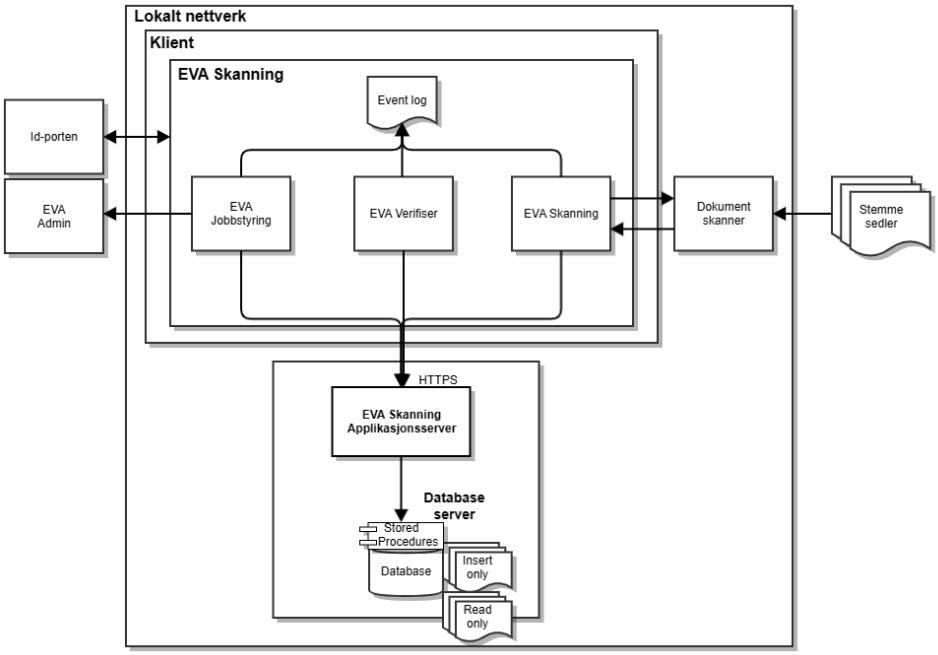
Skisse over oppsett / applikasjonsarkitektur og komponenter

EVA Skannings driftsmiljø er i et lokalt nettverk satt opp hos kommue eller fylkeskommune. Innenfor dette nettverket:

- Settes et gitt antall klientPCer opp
- EVA Skanning installeres på klientPCene
- Dokumentskannere kobles til klientPCene

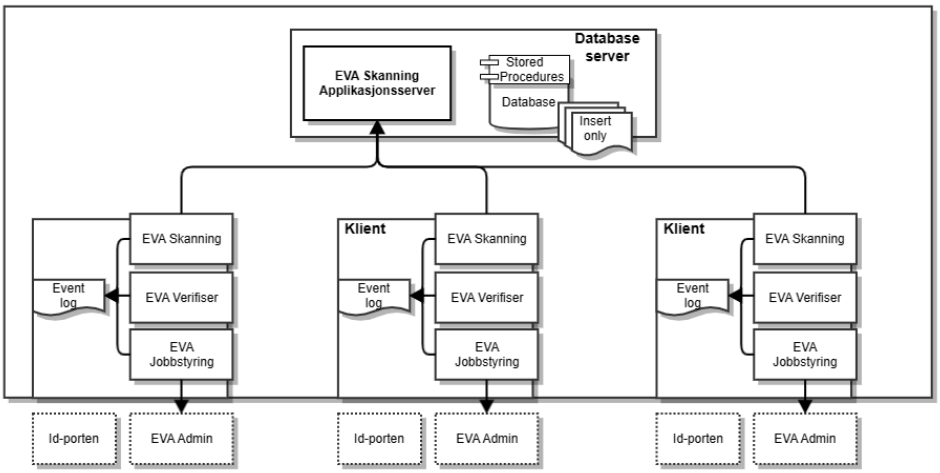
- En database / databaseserver settes opp og EVA Skanning Applikasjonsserver kobles til denne.
  - Kommunikasjon mellom EVA Skanning klientPCer og EVA Skanning Applikasjonsserver krypteres (HTTPS)
- Nettverket åpnes for tilgang til Id-porten for autentisering
- Nettverket åpnes for tilgang til EVA Admin for overføing av telleresultater
- Nettverket åpnes for tilgang til EVA Oppgrader for oppgradering av EVA Skanning

Diagrammet gir en forenklet framstilling av EVA Skanningapplikasjonen som består av 3 moduler, EVA Skanning, EVA Jobbstyring, EVA Verifisering, samt driftskontekst



Skalering

Skalering oppnås som beskrevet ved å legge til flere klienter mot samme applikasjonsserver



Sikring av infrastruktur

EVA Skanning installeres hos kommuner og fylkeskommuner og kjøres således lokalt. En nærmere beskrivelse av sikring av infrastruktur er beskrevet i Sikkerhetsveilederen på valgmedarbeiderportalen. Der det er mulig har Valgdirektoratet inkludert hjelpemidler i installasjonspakken for å bidra til sikring av EVA Skanningklient.

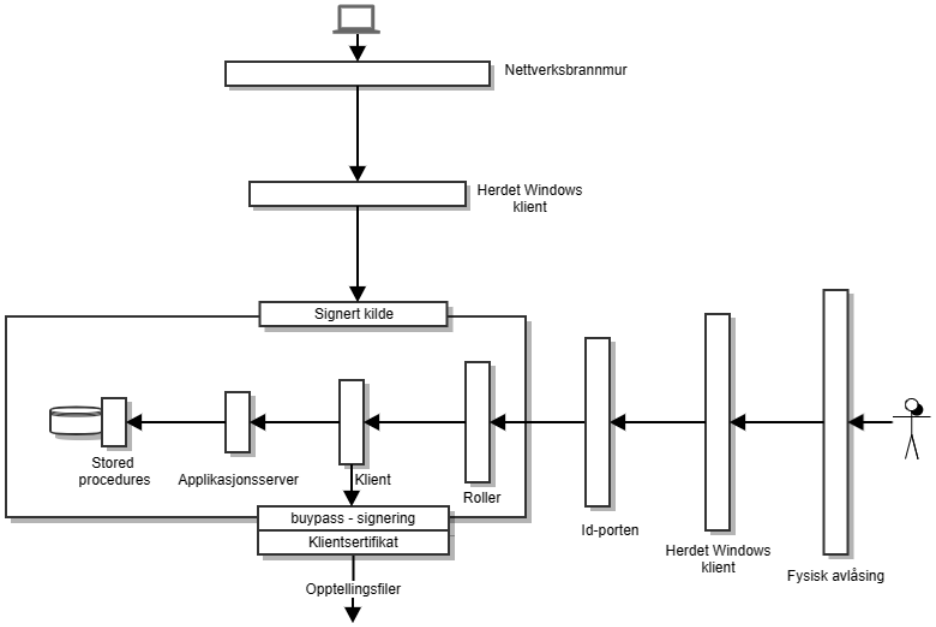
Sikringtiltakene er ikke gjentatt i systemdokumentasjonen utover beskrivelsen av "Lagdeling i EVA Skanning".

Lagdeling i EVA Skanning

EVA Skanning er lagdelt på flere nivåer for på den måten å skape sikkerhetsbarrierer. Prinsippielt skal ikke brudd på en enkelt barriere føre til at systemet ligger åpent.

EVA Skanning er beskyttet gjennom infrastruktur og gjennom hvordan applikasjonen er bygget opp. De forskjellige mekanismene er omtalt mer detaljert i etterfølgende kapiteler.

Skissen viser en forenklet framstilling av barrierene som beskytter EVA Skanning



Logisk tilgang

En bruker som forsøker å aksessere EVA Skanning logisk møter et sett av barrierer:

- Nettverksbrannmur - kun et fåtall definerte adresser er tilgjengelige for kommunikasjon *ut* mot verden fra EVA Skanning
- Windows brannmur - kun et fåtall definerte adresser er tilgjengelige for kommunikasjon *ut* mot verden fra Windowsklienten
- EVA Skannings kommunikasjon med applikasjonsserver er kryptert med HTTPS
- Windows applikasjonsbegrensning - kun EVA Skanningapplikasjoner kan kjøres i Windowsklienten (PCen)
- EVA Skannings kildekode og konfigurasjon er signert med kodesigneringssertifikat

Fysisk tilgang

En bruker som forsøker å aksessere EVA Skanning fysisk møter et sett av barrierer:

- Kun personer med særskilt autorisasjon har tilgang til lokaler hvor systemer kjører
- Lokaler er avlåst med brann og innbruddsalarm
- Bruker må logge inn på Windowsklient (PC) med brukernavn og passord
- Windows applikasjonsbegrensning - kun EVA Skanningapplikasjoner kan kjøres i Windowsklienten (PCen)
- Bruker må autentisere seg gjennom Id-portens autentiseringstjeneste
- Bruker må ha en rolle i EVA Admin for å få tilgang til funksjoner i EVA Skanning
- Bruker må ha utstedt et personlig buypasskort for å kunne signere en telling for overføring til EVA Admin, i tillegg til å inneha et klientsertifikat utstedt av Valgdirektoratet for å oppnå kontakt med EVA Admins oversendelseendepunkt
- Kun administrator og applikasjonsserver har skrive og lesetilgang til databasen

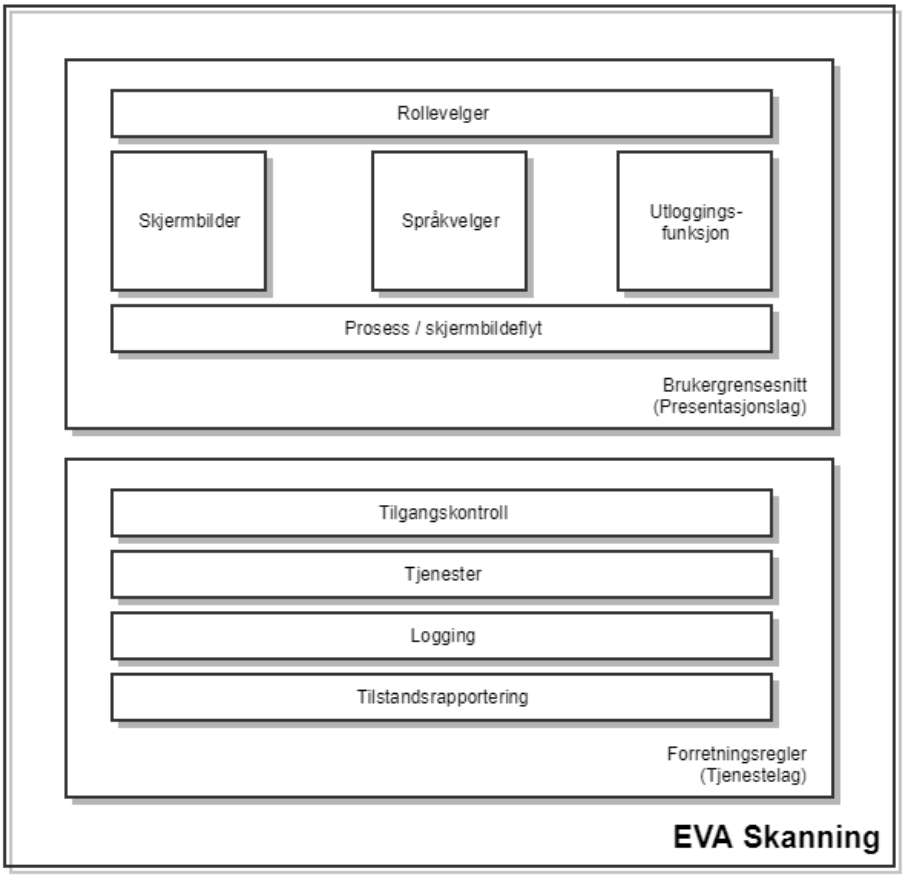
Moduler i EVA Skanning

Eva Skanning applikasjonene kjører alle på samme maskin med lokal database eller database i nettverk.

Alle modulene er bygget opp på samme måte og konseptuelt gir det mening å dele inn disse inn som følger:

- Rollevelger, skjermbilder og prosess/skjermbildeflyt er definert i presentasjonslaget, i samspill med
- Forretningsregler som understøtter opptellingsprosessen i tjenestelaget.

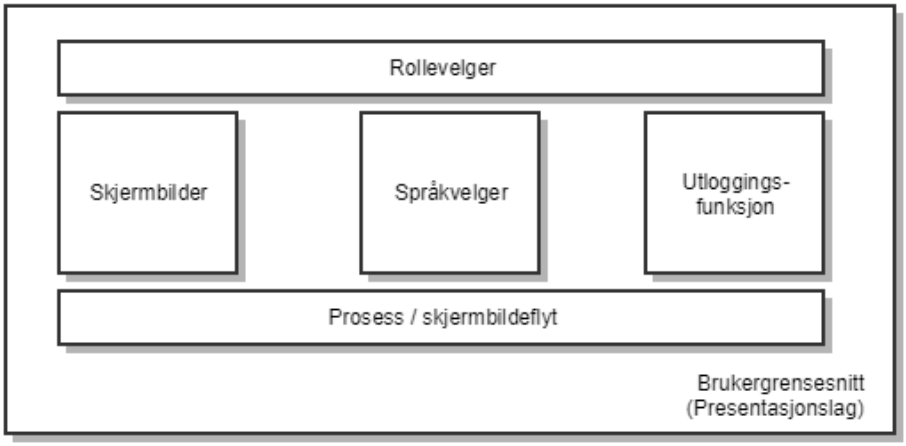
Generell skisse over moduler i EVA Skanning applikasjonene:



Presentasjonslag

Presentasjonslaget er der brukerinteraksjon skjer gjennom brukergrensesnittet.

Skisse over presentasjonslaget



Presentasjonslaget utøver følgende funksjoner:

- Administrerer Id-porten autentisering via EVA Admins tjeneste
- Administrerer rollevalg for innlogget bruker
- Administrerer språkvalg
- Håndhever tilgangskontroll til data i henhold til rollebasert tilgangskontroll
- Presenterer brukergrensesnittet for forretningsprosessene i EVA Skanning
- Håndterer skjermbildeflyt i forretningsprosesser
- Kommuniserer med tjenestelaget for data og forretningsprosesser
- Rapporterer EVA Skannings tilstand til Valgdirektoratets mottakstjeneste for tilstandsrapport

Presentasjonsrammeverk

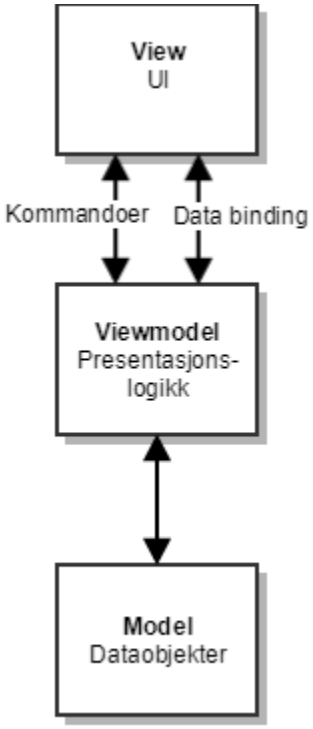
Hensikten med denne beskrivelsen er ikke å gi en utførlig beskrivelse av teknologien, men belyse at/hvordan den brukes i applikasjonene.

Windows Presentation Foundation brukes for å presentere brukergrensesnitt.

MVVM-patternet brukes hovedsaklig for å rendyrke ansvarsområder i presentasjonen:

- View - brukergrensesnitt og brukerinteraksjon
- Model - modellen der data tilgjengeliggjøres (f.eks domenemodellen)
- Viewmodel - Forretningslogikk (lagre, les m.m.), samt synkronisering av view og modell ved hjelp av "databindings" definert i view
- Binder - Knytning mellom view og viewmodel

Forenklet skisse over MVVM



Events og signallering

For å håndtere events og signalering mellom komponenter i klientlaget, benyttes EventAggregator fra Microsoft Prism-bibliotekene. Event Aggregator er en såkalt «weak reference» meldingsbuss, dvs den har som ansvar å formidle meldinger frem og tilbake mellom komponenter, i tillegg til at den under panseret benytter «weak reference» patternet for å holde kontroll på referanser mellom publishers og subscribers. Fordelen med en slik meldingsbuss kontra å bruke det tradisjonelle eventsystemet som er innebygd i CLR, er at man ikke trenger en «hard» referanse til eventen man skal abonnere på meldinger fra. Dette betyr at koden blir mye mindre koblet, mer oversiktlig og mer testbar. EventAggregator fungerer meget bra sammen med Dependency Injection. Det er også en fordel på minnesiden, man trenger i prinsippet ikke tenke på å koble ned eventhandlere før scopet til et objekt går ut. Å bruke EventAggregator eller lignende komponenter er et svært vanlig pattern i rike klientløsninger (og i enhver løsning som har et stateful UI).

Autentisering - id-porten

Autentisering gjøres ved hjelp av EVA Admins autentiseringstjeneste. I skissen over dialog med ID-porten under vil det være EVA Skanning som åpner et integrert nettleservindu og søker å logge inn i EVA Admin. Prinsippielt gjelder følgende (utdrag fra EVA Admin systemdokumentasjon):

Brukere i EVA Admin må autentiseres, dvs. at brukerens identitet må kontrolleres.

Dette gjøres i ID-porten. På denne måten trenger ikke EVA Admin å forvalte brukeridentiteter, men lar i stedet brukerne bruke sin egen, elektroniske ID som f.eks. MinID, BankID eller Buypass.

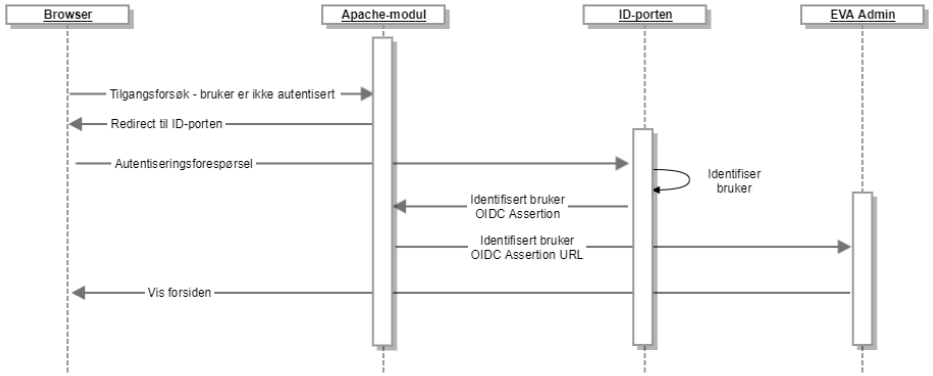
Protokoll

Autentiseringstjenesten bruker OpenID Connect (OIDC)protokollen mot ID-porten

ID-porten dialog

Apache-modulen mod_auth_opendidc brukes som OIDC-megler mot ID-porten.

Forenklet skisse over dialog med ID-porten



- Bruker forespør EVA Admin innloggingside fronend. Apachemodul videresender forespørslene til ID-porten
- ID-porten identifiserer og autentiserer bruker
- Frontend Apachemodul videresender svar på autentiseringsforespørsel til EVA Admins "min side" dersom bruker har en definert rolle i EVA Admin

Autentisering ved oversending av opptellingsfil

Autentiseringstoken som mottas av EVA Skanning ved autentisering via EVA Admin legges ved når opptellingsresultat overføres fra EVA Skanning til EVA Admin.

EVA Admin vil kontrollere om bruker har en gyldig sesjon i EVA Admin, dersom ikke vil forespørsel om overføring avvises.

Autorisering - Rollebasert tilgangskontroll

EVA Skanning håndhever en forenklet versjon av rollebesert tilgangskontroll (RBAC), besert på samme konsept som i EVA Admin.

EVA Skanning operer kun med 2 roller:

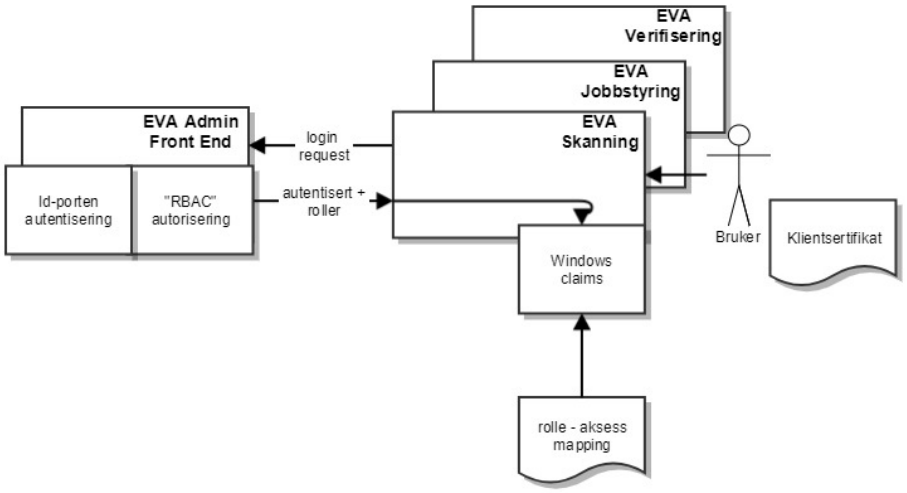
- Valgansvarlig
- Skanningoperatør.

Brukerens rolle hentes fra EVA Admin ved autentisering og håndheves videre av EVA Skanningmodulene.

Håndheving av tilganger for rollene er en integrert del av programtilgangen.

Se for øvrig systemdokumentasjon for EVA Admin for detaljer rundt rollemodell "Autorisering - Rollebasert tilgangskontroll (RBAC)".

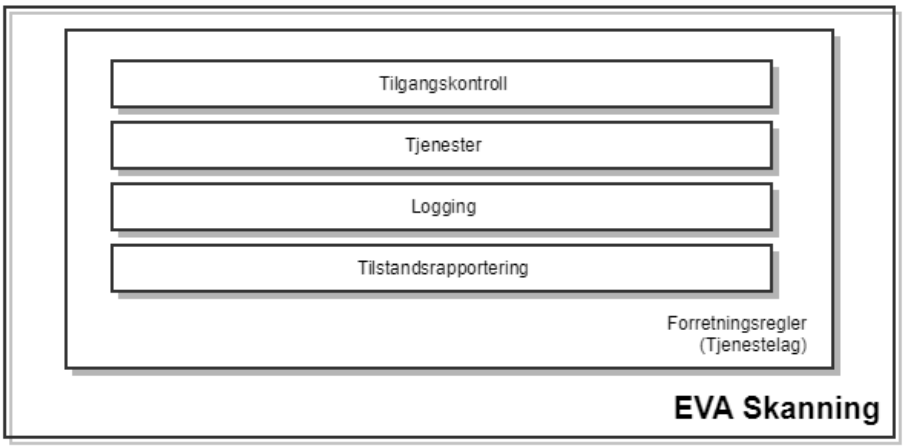
Skissen illustrerer en forenklet framstilling av rolleautorisasjon i EVA Skanning:



Tjenestelag

Tjenestelaget er modulen der forretningsregler håndheves og data formidles mellom presentasjonslag og database.

Skisse over tjenestelaget



Tjenestelaget utøver følgende funksjoner:

- Utfører forretningsprosesser på forespørsler fra presentasjonslaget
- Håndhever forretningsregler for uthenting og oppdatering av data fra presentasjonslaget mot databasen
- Formidler forespørsler om data fra presentasjonslaget til databasen
- Formidler forespørsler om oppdatering av data fra presentasjonslaget til databasen
- Utfører logging av forretningsprosesser

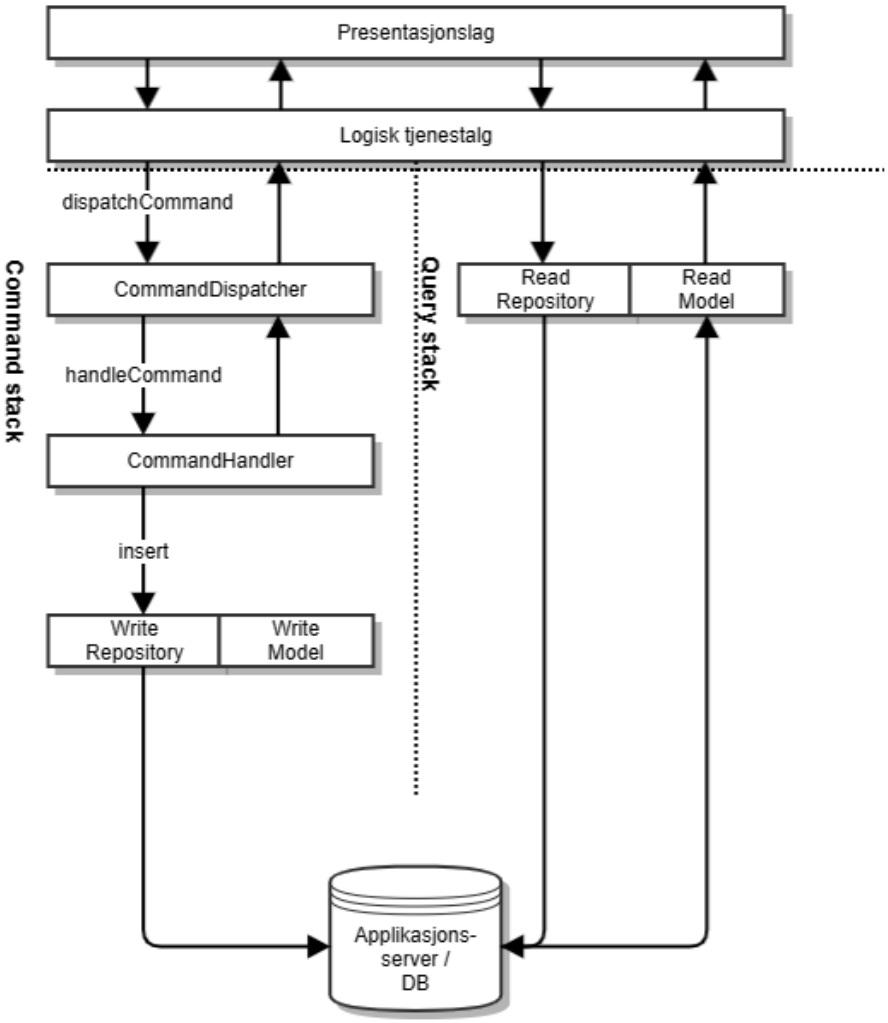
Command Query Responsibility Segregation

De tre EVA Skanning-applikasjonene er i hovedsak like når det kommer til valg av komponenter, rammeverk og design patterns:

- I klientene brukes MVVM-pattern med databinding mot WPF-views.
- På tjenestelaget og mot databasen er det et CQRS-pattern som er benyttet, med separasjon av lese- og skrivemodeller mot databasen

Mot databasen eksekveres stored procedures gjennom mikro-ORMen Dapper. Siden systemet har høy last i større skannsenter kreves det god kontroll på hvilke spørringer som eksekveres.

Skissen er en konseptuell modell av arkitektur i tjenestelag og nedover



Av skissen kan man se de to hovedaksene i tjenestelagsarkitekturen - kommandosiden og spørresiden.

Kommandosiden har ansvar for all skriving til databasen, dette skjer i CommandHandler, som kaller forskjellige metoder i skriverepositoriene for igjen å kjøre de forskjellige databasekommandoene mot databasen.

Ett og bare ett transaction scope skal eksistere i en enkelt command handler, slik at alt som skjer inni en command handler har transaksjonell integritet.

I det man i en command handler begynner å manipulere på data, skjer det låsing i databasen. Siden man konkret åpner en direkte connection mot databasen fra hver enkelt klient, så er også nettverkslatency noe som bidrar til å øke tiden en tabell er låst. Command handlere vil derfor være svært «to the point» og gjennomtenkt.

En command handler kan i prinsippet returnere data synkront, for eksempel den nylig innsatte primærnøkkelverdien for en av tabellene man legger til data i. Systemet er å betrakte som et synkront system, der man blokkerer/venter på svar.

Spørresiden håndterer alle spørringer, og her er rene leseoperasjoner som ofte gjør aggregeringer og summeringer.

Hvert «lag» i arkitekturen har i utgangspunktet sine egne modeller og DTOer.

Ved å opprettholde logiske skiller og nivåer i systemet får man et system som blir mer endringsdyktig. Systemet er et distribuert system, men det er ingen .NET-kode som kjører sentralt, kun database.

Dapper - Micro ORM

Dapper som er et lettvekts ORM rammeverk brukes som ORM, i praksis for å håndtere stored procedures kall mot databasen og presentere resultater i en objektmodell.

Gitt bruken av stored procedures er det ikke behov for fullskala ORM-rammeverk med objektrelasjonell mapping og databaseinterasjon. Bruken av CQRS-patternet med separate lese- og skrivemodell gjør også Dapper til et egnet ORM-rammeverk.

Dependency injection

Prinsipper som Inversion of Control og Dependency Injection er svært sentrale i EVA Skannings arkitektur. Dette betyr at man har en deklarativ tilnærming til å nyttiggjøre seg andre klasser og komponenter i systemet. Dette oppnås vha constructor injection, og en Inversion of Control container som fasiliteter og sørger for at dette går smidig. Systemet bruker Unity IoC container.

Logging

Det gjøres teknisk logging til event logg i Windows. Logging gjøres på 4 nivåer:

- Info
- Warning
- Error
- Critical

Tilstandsrapportering

EVA Skanning vil ved oppstart rapportere status på sikkerhetstiltak på klient-PC til et endepunkt hos Valgdirektoratet.

Tilstandsrapporten inneholder blant annet:

- Hvorvidt Windows Firewall er aktivisert
- Hvorvidt Bitlocker er aktivisert
- Hvorvidt database "connect string" er kryptert.

Rapporteringen gjøres primært for å måle effekten av Valgdirektoratets sikkerhetsveiledere, ansvaret for gjennomføring av sikkerhetstiltak ligger hos den enkelte kommune og fylkeskommune.

Database

EVA Skanning applikasjonene bruker Microsoft SQL databaseserver/database for lagring av data. Databasen er en standard relasjonsdatabase.

Databaseskjema

Databasen består av ett databaseskjema for maskinell telling av stemmedler.

Typer definert er:

- Tables - tabeller
- Views - views (denormaliseringer)
- stored prcedures - lagrede prosedyrer
- function - funksjoner
- autonumber - funksjon for å generere unike primærnøkler

Samtlige forekomster i tabeller identifiseres ved:

- Id (faktisk primærnøkkel)

For nærmere beskrivelse av relasjons- og databasemodell se domenemodell.

Endringssporing

Oppdateringsprinsippet for data i databasen er "insert only", dette betyr at endringssporing ivaretas ved bruk av historikk.

Kodeorganisering - prinsipper og retningslinjer

For beskrivelse av kodeorganisering, se bekrivelser av MVVM og CQRS i tidligere kapitler.

Kodeeksempel - MVVM / CQRS

Koden under eksemplifiserer MVVM og CQRS patterns som brukes i EVA Skanning modulene.

Komplett kode finnes i kildekoden, eksempelet kan være noe forenklet for å illustrere flyten fra brukergrensesnitt til database.

MVVM - View

View for sletting av en telling er definer i xaml, her ligger også knytning mot viewModel (counting:DeleteCountingPopupViewModel) og kommando (DeleteCountingCommand)

View

```
<userInterfaceElements:EvaWindow x:Class="Valg.EVA.Skanning.JobManagement.Startup.UI.Counting.
DeleteCountingPopupView"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:counting="clr-namespace:Valg.EVA.Skanning.JobManagement.Startup.UI.Counting"
        xmlns:userInterfaceElements="clr-namespace:Valg.EVA.Skanning.UserInterfaceElements;assembly=Valg.EVA.
Skanning.UserInterfaceElements"
        mc:Ignorable="d"
        Style="{DynamicResource PopUpWindowAction}"
        WindowStartupLocation="CenterOwner"
        ShowInTaskbar="False"
        d:DataContext="{d:DesignInstance d:Type=counting:DeleteCountingPopupViewModel}"
        Title="{Binding Title}">
    <Window.Resources>
        <ResourceDictionary>
            <Style TargetType="Button" BasedOn="{StaticResource LinkButtonNormal}">
                <Setter Property="Margin" Value="0,0,0,5" />
            </Style>
        </ResourceDictionary>
    </Window.Resources>
    <StackPanel Margin="30" Height="auto" Width="auto">
        <DockPanel Margin="0,0,0,20">
            <TextBlock Text="{Binding Title}" HorizontalAlignment="Left" Style="{StaticResource
TextBlockMediumHeader}" />
        </DockPanel>
        <Button Content="{Binding JobManagementLanguageString.DeleteCountingPopup_DeleteCountingText}" Command="
{Binding DeleteCountingCommand}" />
        <Button Content="{Binding JobManagementLanguageString.DeleteCountingPopup_AbortText}" Margin="0,20,0,0"
IsCancel="True" />
    </StackPanel>
</userInterfaceElements:EvaWindow>
```

MVVM - kommando

kommando for slett telling

command

```
using System;

namespace Valg.EVA.Skanning.Commands.DeleteCounting
{
    public class DeleteCountingCommand : ICommand
    {
        private readonly int countingId;
        private readonly DateTime countingConcurrencyToken;
        private readonly string modifiedBy;

        public DeleteCountingCommand(int countingId, DateTime countingConcurrencyToken, string modifiedBy)
        {
            this.countingId = countingId;
            this.countingConcurrencyToken = countingConcurrencyToken;
            this.modifiedBy = modifiedBy;
        }

        public int CountingId
        {
            get { return countingId; }
        }
        public DateTime CountingConcurrencyToken
        {
            get { return countingConcurrencyToken; }
        }
        public string ModifiedBy
        {
            get { return modifiedBy; }
        }
    }
}
```

MVVM - Viewmodel

Viewmodelen inneholder knytning til modell og tjenestelag.

Viewmodel

```
using System;
using Prism.Commands;

namespace Valg.EVA.Skanning.JobManagement.Startup.UI.Counting
{
    public class DeleteCountingPopupViewModel : ReactiveViewModel
    {
        private readonly IJobManagementApplication jobManagementApplication;

        public DeleteCountingPopupViewModel(string countingName, Data.CountingModel countingModel,
IJobManagementApplication jobManagementApplication)
        {
            CountingName = countingName;
            CountingModel = countingModel;
            this.jobManagementApplication = jobManagementApplication;

            DeleteCountingCommand = new DelegateCommand(DeleteCountingHandler);
        }

        public Data.CountingModel CountingModel { get; private set; }

        public string CountingName { get; private set; }

        public DelegateCommand DeleteCountingCommand { get; private set; }

        public Action CloseWindowAction { get; set; }

        public string Title { get { return string.Format(Valg.EVA.Skanning.Resources.Language.
JobManagementLanguageString.DeleteCountingPopup_Title, CountingName); } }

        private void DeleteCountingHandler()
        {
            jobManagementApplication.DeleteCounting(CountingModel);
            CloseWindowAction();
        }
    }
}
```

MVVM - model

Modellen definerer dataobjekt for telling

Model

```
using System;
using Valg.EVA.Skanning.Contracts.CommonEnums;
using Valg.EVA.Skanning.Resources;
using Valg.EVA.Skanning.Resources.Language;
using ReactiveUI;

namespace Valg.EVA.Skanning.JobManagement.Startup.Data
{
    public class CountingModel : ReactiveObject, IUpdatable<CountingModel, int>
    {
        private int id;
        public int Id
        {
            get { return id; }
            set { this.RaiseAndSetIfChanged(ref id, value); }
        }

        public string Name
        {
```

```
    get
    {
        return string.Format(
            "{0} {1}",
            LanguageString.ResourceManager.GetEnumString(CountingType),
            CountNumber);
    }
}

private int countNumber;
public int CountNumber
{
    get { return countNumber; }
    set { this.RaiseAndSetIfChanged(ref countNumber, value); }
}

private CountingType countingType;
public CountingType CountingType
{
    get { return countingType; }
    set { this.RaiseAndSetIfChanged(ref countingType, value); }
}

private CountingState countingState;
public CountingState CountingState
{
    get { return countingState; }
    set { this.RaiseAndSetIfChanged(ref countingState, value); }
}

private string uniqueId;
public string UniqueId
{
    get { return uniqueId; }
    set { this.RaiseAndSetIfChanged(ref uniqueId, value); }
}

private string contestId;
public string ContestId
{
    get { return contestId; }
    set { this.RaiseAndSetIfChanged(ref contestId, value); }
}

private string electionEventId;
public string ElectionEventId
{
    get { return electionEventId; }
    set { this.RaiseAndSetIfChanged(ref electionEventId, value); }
}

private string emlAreaContextId;
public string EmlAreaContextId
{
    get
    {
        return emlAreaContextId;
    }
    set { this.RaiseAndSetIfChanged(ref emlAreaContextId, value); }
}

private string countingEmlAreaContextId;
public string CountingEmlAreaContextId
{
    get
    {
        return countingEmlAreaContextId;
    }
    set { this.RaiseAndSetIfChanged(ref countingEmlAreaContextId, value); }
}
```

```
        private string votingCategory;
        public string VotingCategory
        {
            get { return votingCategory; }
            set { this.RaiseAndSetIfChanged(ref votingCategory, value); }
        }

        private string electionIdentifier;
        public string ElectionIdentifier
        {
            get { return electionIdentifier; }
            set { this.RaiseAndSetIfChanged(ref electionIdentifier, value); }
        }

        private string electionGroup;
        public string ElectionGroup
        {
            get { return electionGroup; }
            set { this.RaiseAndSetIfChanged(ref electionGroup, value); }
        }

        public bool IsFinalCounting
        {
            get { return CountingType == CountingType.Final; }
        }

        public bool IsCountingComplete
        {
            get { return CountingState == CountingState.Complete; }
        }
        private DateTime concurrencyToken;
        public DateTime ConcurrencyToken
        {
            get { return concurrencyToken; }
            set { this.RaiseAndSetIfChanged(ref concurrencyToken, value); }
        }

        public void Update(CountingModel updatedModel)
        {
            CountingState = updatedModel.CountingState;
            ConcurrencyToken = updatedModel.ConcurrencyToken;
        }
    }
}
```

CQRS - Service for telling (Logisk tjenestelag)

Service for sletting av tellinger

Service

```
using System;
using Valg.EVA.Skanning.Commands;
using Valg.EVA.Skanning.Commands.DeleteCounting;
using Valg.EVA.Skanning.Commands.RegisterBoxAsEmpty;
using Valg.EVA.Skanning.Repositories.Counting;
using Valg.EVA.Skanning.ServicesModel.Internal;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Box.Write.EmptyBox;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Count.Read;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Count.Write;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using Valg.EVA.Skanning.Contracts;
using Valg.EVA.Skanning.Contracts.CommonEnums;
using Valg.EVA.Skanning.Commands.DeleteBoxCountingAndRegisterForScanAgain;
using Valg.EVA.Skanning.Commands.ResolveBoxCountingDuplicates;
using Valg.EVA.Skanning.Commands.ResolveLockedBallots;
using Valg.EVA.Skanning.Commands.StartCounting;
using Valg.EVA.Skanning.Diagnostics;
using Valg.EVA.Skanning.Repositories.Box;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Box.Read;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Box.Write;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.JobManage.Write;
using Valg.EVA.Skanning.ServicesModel.ServiceContracts.Scan.Write;
using BoxCountingDetails = Valg.EVA.Skanning.ServicesModel.ServiceContracts.Count.Read.BoxCountingDetails;
using Counting = Valg.EVA.Skanning.ServicesModel.ServiceContracts.Count.Read.Counting;

namespace Valg.EVA.Skanning.Services.Count
{
    public class CountingService : ICountingService
    {
        private readonly ICommandDispatcher commandDispatcher;
        private readonly ICountingRepository countingRepository;
        private readonly IBoxRepository boxRepository;

        public CountingService(
            ICommandDispatcher commandDispatcher,
            ICountingRepository countingRepository,
            IBoxRepository boxRepository)
        {
            this.commandDispatcher = commandDispatcher;
            this.countingRepository = countingRepository;
            this.boxRepository = boxRepository;
        }


        public DeleteCountingResult DeleteCounting(DeleteCountingParameters parameters)
        {
            var command = new DeleteCountingCommand(
                parameters.CountingId,
                parameters.CountingConcurrencyToken,
                parameters.ModifiedBy);

            var commandResult = commandDispatcher.DispatchCommand<DeleteCountingCommand,
DeleteCountingCommandResult>(
                command);

            return new DeleteCountingResult(commandResult.DeleteFailed);
        }
    }
}
```

CQRS - commandDispatcher

Commanddispatcher er generisk

Dispatcher

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Unity;

namespace Valg.EVA.Skanning.Commands
{
    public class CommandDispatcher : ICommandDispatcher
    {
        private readonly IUnityContainer container;

        public CommandDispatcher(IUnityContainer container)
        {
            this.container = container;
        }

        public void DispatchCommand<T>(T command)
        {
            var commandHandler = ResolveHandlerForCommandType<T>(command.GetType());
            commandHandler.HandleCommand(command);
        }

        public async Task DispatchCommandAsync<T>(T command)
        {
            var commandHandler = ResolveAsyncHandlerForCommandType<T>(command.GetType());
            await commandHandler.HandleCommandAsync(command).ConfigureAwait(false);
        }

        public async Task<R> DispatchCommandAsync<T, R>(T command)
        {
            var commandHandler = ResolveAsyncHandlerForCommandType<T, R>(command.GetType());
            var result = await commandHandler.HandleCommandAsync(command).ConfigureAwait(false);
            return result;
        }

        private IHandleAsyncCommandsWithResult<T, R> ResolveAsyncHandlerForCommandType<T, R>(Type commandType)
        {
            var handleCommandsType = typeof(IHandleAsyncCommandsWithResult<T, R>);
            var handler = ResolveHandler(commandType, handleCommandsType);
            return (IHandleAsyncCommandsWithResult<T, R>)handler;
        }

        private IHandleAsyncCommands<T> ResolveAsyncHandlerForCommandType<T>(Type commandType)
        {
            var handleCommandsType = typeof(IHandleAsyncCommands<T>);
            var handler = ResolveHandler(commandType, handleCommandsType);
            return (IHandleAsyncCommands<T>)handler;
        }

        public R DispatchCommand<T, R>(T command)
        {
            var commandHandler = ResolveHandlerForCommandType<T, R>(command.GetType());
            commandHandler.HandleCommand(command);
            return commandHandler.CommandResult;
        }

        private IHandleCommandsWithReturnValue<T, R> ResolveHandlerForCommandType<T, R>(Type commandType)
        {
            var handleCommandsType = typeof(IHandleCommandsWithReturnValue<T, R>);
            var handler = ResolveHandler(commandType, handleCommandsType);
            return (IHandleCommandsWithReturnValue<T, R>)handler;
        }

        private IHandleCommands<T> ResolveHandlerForCommandType<T>(Type commandType)
```

```
        {
            var handleCommandsType = typeof(IHandleCommands<T>);
            var handler = ResolveHandler(commandType, handleCommandsType);
            return (IHandleCommands<T>)handler;
        }

        private object ResolveHandler(Type commandType, Type handleCommandsType)
        {
            var registration = container.Registrations.Where(c =>
            {
                if (handleCommandsType.IsAssignableFrom(c.MappedToType))
                {
                    var interfaces = c.MappedToType.GetInterfaces();
                    return interfaces.Any(i => i.GenericTypeArguments[0] == commandType);
                }

                return false;
            }).First();
            var handler = container.Resolve(registration.RegisteredType);
            return handler;
        }
    }
}
```

CQRS - commandHandler

CoammandHandleren definerer gjennmføringen av sletting og forretningsregler knyttet til sletting av telling

CommandHandler

```
using System.Data;
using Valg.EVA.Skanning.Diagnostics;
using Valg.EVA.Skanning.Entity.SqlConnection;
using Valg.EVA.Skanning.Repositories.Counting;

namespace Valg.EVA.Skanning.Commands.DeleteCounting
{
    public class DeleteCountingCommandHandler : IHandleCommandsWithReturnValue<DeleteCountingCommand,
DeleteCountingCommandResult>
    {
        private readonly ICountingWriteableRepository countingWriteableRepository;
        private readonly TransactionScopeFactory transactionScopeFactory;

        public DeleteCountingCommandHandler(ICountingWriteableRepository countingWriteableRepository,
TransactionScopeFactory transactionScopeFactory)
        {
            this.countingWriteableRepository = countingWriteableRepository;
            this.transactionScopeFactory = transactionScopeFactory;
        }

        public void HandleCommand(DeleteCountingCommand command)
        {
            CommandResult = new DeleteCountingCommandResult();

            using (var scope = transactionScopeFactory.GetTransactionScope())
            {
                try
                {
                    countingWriteableRepository.MarkCountingAsDeleted(command.CountingId, command.
CountingConcurrencyToken, command.ModifiedBy);

                    scope.Complete();
                }
                catch (DBConcurrencyException ex)
                {
                    EvaSkanningEventSource.Log.ConcurrencyError(GetType().ToString(), ex.ToString());
                    CommandResult.DeleteFailed = true;
                }
            }
        }

        public DeleteCountingCommandResult CommandResult { get; private set; }
    }
}
```

CQRS - Write Repository Interface

Interface

```
using System;
using Valg.EVA.Skanning.Contracts.CommonEnums;
using Valg.EVA.Skanning.Entity.WriteModel;

namespace Valg.EVA.Skanning.Repositories.Counting
{
    public interface ICountingWriteableRepository
    {
        void MarkBoxCountingAsDeleted(int countingId, string barcodeLong, string barcodeShort, DateTime
lastModified);
    }
}
```

CQRS - Write Repository

Repository besørger kommunikasjon med databasen gjennom stored procedures

Write repository

```
using Valg.EVA.Skanning.Contracts.CommonEnums;
using Valg.EVA.Skanning.Entity.SqlConnection;
using Valg.EVA.Skanning.Entity.WriteModel;
using System;

namespace Valg.EVA.Skanning.Repositories.Counting
{
    public class CountingWriteableRepository : ICountingWriteableRepository
    {
        private readonly ISqlConnectionFactory connectionFactory;

        public CountingWriteableRepository(ISqlConnectionFactory connectionFactory)
        {
            this.connectionFactory = connectionFactory;
        }


        public void MarkCountingAsDeleted(int countingId, DateTime lastModified, string modifiedBy)
        {
            connectionFactory.SP_ExecuteVoid(
                "Counting_MarkCountingAsDeleted",
                new
                {
                    CountingId = countingId,
                    LastModified = lastModified,
                    ModifiedBy = modifiedBy
                });
        }
    }
}
```

CQRS - Write Repository - database connection

Infrastrukturlag for write repository

Database connection

```
using Dapper;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;

namespace Valg.EVA.Skanning.Entity.SqlConnection
{
        public static void SP_ExecuteVoid(this ISqlConnectionFactory factory, string spName, object parameters
= null, int? timeOut = defaultCommandTimeOut)
        {
            ExecuteAndCheckForConcurrencyExceptions(() =>
            {
                using (var connection = factory.CreateSqlConnection())
                {
                    connection.Open();
                    connection.Execute(spName, parameters, commandType: CommandType.StoredProcedure,
commandTimeout: factory.GetCommandTimeOut(timeOut));
                }
            });
        }
    }
}
```

Integrasjoner

Id-porten - brukerautentisering

Autentisering gjøres ved hjelp av EVA Admins autentiseringstjeneste. I skissen over dialog med ID-porten under vil det være EVA Skanning som åpner et integrert nettleservindu og søker å logge inn i EVA Admin. Prinsippielt gjelder følgende (utdrag fra EVA Admin systemdokumentasjon):

Brukere i EVA Admin må autentiseres, dvs. at brukerens identitet må kontrolleres.

Dette gjøres i ID-porten. På denne måten trenger ikke EVA Admin å forvalte brukeridentiteter, men lar i stedet brukerne bruke sin egen, elektroniske ID som f.eks. MinID, BankID eller Buypass.

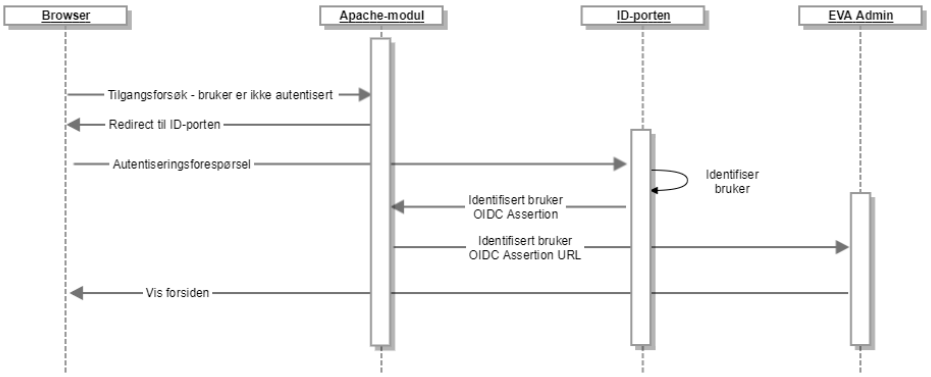
Protokoll

Autentiseringstjenesten bruker OpenID Connect (OIDC)protokollen mot ID-porten

ID-porten dialog

Apache-modulen mod_auth_opendidc brukes som OIDC-megler mot ID-porten.

Forenklet skisse over dialog med ID-porten



- Bruker forespør EVA Admin innloggingside fronend. Apachemodul videresender forespørslene til ID-porten
- ID-porten identifiserer og autentiserer bruker
- Frontend Apachemodul videresender svar på autentiseringsforespørsel til EVA Admins "min side" dersom bruker har en definert rolle i EVA Admin

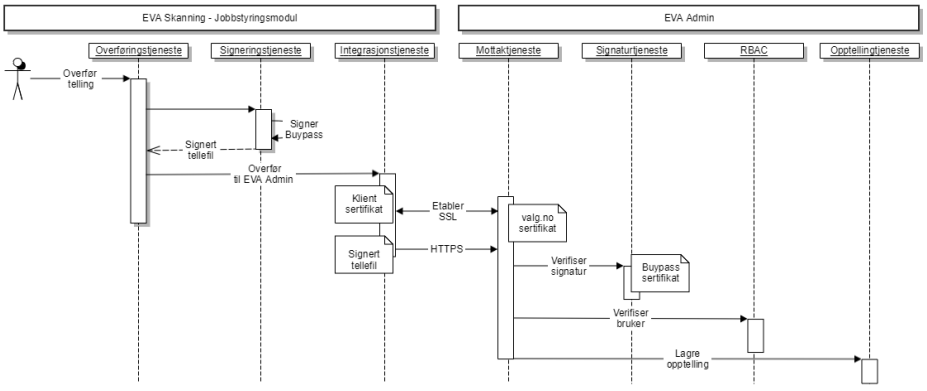
EVA Admin - oversending av telleresultater

Ved ferdigstilling av en gitt telling skal telleresultatet oversendes til EVA Admin.

Overføringen av opptellingsfilen sikres på flere måter:

- Ved roller: Det er kun valgansvarlig som kan overføre opptellingsfiler fra EVA Skanning til EVA Admin
- Ved signering av telleresultat: Valgansvarlig må signere tellefilen ved hjelp av Buypass signeringstjeneste med Buypasskort
- Ved tilgangskontroll: EVA Skanning må inneha et klientsertifikat for å få tillatelse til å kommunisere med EVA Admin
- Ved kryptering: Tellefilen oversendes over HTTPS med asymmetrisk kryptering
- Ved verifisering i EVA Admin: Ved mottak av tellefil vil EVA Admin verifisere at avsenders signatur har opphav i forventet Buypass-sertifikat
- Ved autentisering: EVA Admin: Ved mottak av tellefil vil EVA Admin kontrollere om signatur fra avsender stemmer overens med bruker som er definert som valgansvarlig i EVA Admin

Skissen illustrer en forenklet framstilling av oversending av opptellingsresultat fra EVA Skanning (jobbstyringsmodulen) til EVA Admin med fokus på sikkerhetsmekanismer (skissen inneholder begge systemer for å sikre helheten)



Tilstandsrapportering til Valgdirektoratet

EVA Skanning vil ved oppstart rapportere status på sikkerhetstiltak på klient-PC til et endepunkt hos Valgdirektoratet.

Tilstandsrapporten inneholder blant annet:

- Hvorvidt Windows Firewall er aktivisert
- Hvorvidt Bitlocker er aktivisert

- Hvorvidt database "connect string" er kryptert.

Rapporteringen gjøres primært for å måle effekten av Valgdirektoratets sikkerhetsveiledere, ansvaret for gjennomføring av sikkerhetstiltak ligger hos den enkelte kommune og fylkeskommune.

Vedlegg

Systemkrav

PC

Maskinvare	Beskrivelse
Bærbare PC med internettilgang	minimum: ▪ 16 GB internminne ▪ Intel Core i7 prosessor ▪ 250 GB SSD disk ▪ 15" tommers skjerm – oppløsning 1920x1080 ▪ 4 USB porter
Operativsystem	▪ Windows 10 Enterprise (64-bit)
Nødvendig tilbehør	• Innebygget smartkortleser for signering av opptellinger ved overføring til EVA Admin eller • USB smartkortleser for signering av opptellinger ved overføring til EVA Admin • Håndholdt USB strekkodeskanner
Anbefalt tilbehør	▪ USB-mus ▪ Ekstern skjerm
Nødvendig programvare	▪ EVA Skanning ▪ Browser (Edge, Chrome eller tilsvarende)

Forklaringer:

HID: Human Interface Device – strekkodeleseren skal fungere som forlengelse av tastaturet slik at strekkoden leser inn tall i felt for manntallsnummer. HID produkter er merket med forkortelsen HID eller Human interface Device.

Database

Maskinvare

Dette er veldig avhengig av hvor mange stemmeberettigede du har, og hvor mange klienter (jobbstyring, skann, verifiser og stikkprøve) du skal kjøre samtidig.

Det som påvirker ytelsen mest er lese/skrive -hastigheten på harddisken.

Databaseserveren må være dedikert for valggjennomføringen og ikke brukes til andre formål.

Som et minimum skannsenter på 1 jobbstyrer, 2 skannere og 2 verifiserere anbefaler vi minimum:

- 4 kjerner dedikert CPU
- 16 GB RAM
- SSD disk - for størrelse kan du følge beregning nedenfor
- gigabit ethernet

Stor installasjon - skannsenter	
Database	Microsoft SQL Server 2019 eller nyere (standard edition eller høyere)
Operativsystem	Windows Server 2019 eller 2022 (standard edition eller høyere)

LIten installasjon - enkeltPC med skanner tilkoblet	
Datasbase	Microsoft SQL Express installeres på samme maskin som EVA Skanning og har derfor samme systemkrav som klientPC for EVA Skanning

Valgdirektoratet anbefaler at du tar kontakt med egen IT-ansvarlig for spørsmål rundt anbefalingene.

Anslag av stemmeseddelestørrelse

Størrelsen på databasen avhenger av om man velger å lagre sedler i farger eller sort-hvitt

Antall	Farger	Sort-hvitt
1	3 MB	200 kB
1 000	3 GB	200 MB
10 000	31 GB	2 GB
100 000	314 GB	20 GB
1 000 000	3,2 TB	200 GB

Nettverkstrafikk

Nettverkstrafikken avhenger av hvor mange skannere og verifiseringsstasjoner som brukes til en hver tid.

Skanning (her antas det at det skannes ca. en seddel i sekundet)

Antall Scannere	Farger	Sort-hvitt
1	24 Mb/s	1,6 Mb/s
2	48 Mb/s	3,2 Mb/s
5	120 Mb/s	8 Mb/s
10	240 Mb/s	16 Mb/s
20	480 Mb/s	32 Mb/s

Verifisering (her antas det at man bruker ca 4 sekunder på å verifisere en seddel)

Antall Scannere	Farger	Sort-hvitt
1	6 Mb/s	0,4 Mb/s
2	12 Mb/s	0,8 Mb/s
5	30 Mb/s	2 Mb/s
10	60 Mb/s	4 Mb/s
20	120 Mb/s	8 Mb/s

Dokumentskanner

Tabell over anbefalte dokumentskannermodeller til bruk sammen med EVA Skanning ved valget i 2023

Produsent	Modell	PCI Vendor Id	PCI Product Id
Canon	DR-G2110	1083	166F
Canon	DR-G2140	1083	166E
Canon	DR-G2090		

Valgdirektoratet *kan ikke garantere at andre skannermodellers vil støttes av EVA Skanning.*

Tekniske krav til skannermodeller

Skannermodeller beskrevet i tabellen over tilfredstiller kravene beskrevet under.

Generelle krav:

- Type: Tosidig A3-skrivebordsskanner med arkmater
- Oppløsning: 600 dpi
- Lyskilde: RGB LED
- Skannerside: Enkeltsidig, tosidig
- Grensesnitt: USB 2.0 eller bedre
- Strømkilde: AC 220 – 240V
- Lyd: 60 db
- Støtte for RGB «drop out» farge
- Miljøsamsvar: EPEAT Gold sertifisert / ENERGY STAR®, RoHS eller tilsvarende

Skanningmoduser:

- farge, gråskala, sort-hvitt
- Mulighet for separat innstilling av lys/kontrast/farge på for- og bakside
- Forskyvningskorrigering
- «Multistream» - mulighet for å hente flere varianter av samme bilde (sort/hvitt – farge)

Kapasitet / hastighet:

- Automatisk arkmating
- Sort-hvitt-skanning: A4 landskap - 100 sider pr minutt (ppm) / 300 dpi
- Fargeskanning: A4 landskap - 100 sider pr minutt (ppm) / 300 dpi
- Innmatningskapasitet: 500 ark. For modell DR-G2090: 300 ark
- Dobbeltmatningshindring
- Forventet daglig driftssyklus: 50 000 skanninger pr dag. For modell DR-G2090: 30 000 skanninger pr. dag

Operativsystem / drivere

- Windows 10 (Enterprise)
- Windows ISIS driver 32/64 bit

Forbruksutstyr:

- «Roller kit»
- Renseutstyr