Kartverket

Innholdsfortegnelse

Dokumentasjon matrikkelAPI

Dokumentasjon matrikkelAPI

Introduksjon

MatrikkelAPI-et er et webtjeneste-API laget for tilgang til matrikkeldata, både i små og store mengder. Det finnes tjenester både for å søke opp data basert på logiske inndata, som adressenavn, husnummer og bokstav for en adresse, men også for å søke opp data basert på numeriske id-er.

Denne dokumentasjonen skal gi et innblikk i hvordan API-et er bygget opp og hvordan man kan bruke det. Det vil gi innsikt i hvordan man går fram for å ta API-et i bruk, hvordan man gjør søk og hvordan man navigerer i API-ets modell.

Tidsplan for utvikling og release

3. kvartal 2016 tilgjengelig i test 1. kvartal 2017 produksjon matrikkel	Innsynstjenester
1. kvartal 2017 tilgjengelig i test 3. kvartal 2017 produksjon matrikkel	Endringsloggtjenester
1.kvartal 2018 produksjon matrikkel	Oppdateringstjenester

Planen for utlegging av tjenester i det nye matrikkelAPI-et er som beskrevet over. Innsynstjenestene er i produksjon fra 1. kvartal 2017 og vi følger så på med endringsloggstjenester og oppdateringstjenester fortløpende.

Oppsett av dokumentasjon

Dokumentasjonen er delt inn i kapitler med forskjellige fokus. Vi starter med et kapittel for hvordan man kan laste ned `wsdl`-er og `xsd`-er for API-et. Etter dette kommer introduksjon til boblemodellen brukt i matrikkelen og API-et samt navigasjon i modellen. Etter dette er det laget et eget kapittel for å beskrive historikk i matrikkelen da dette er et sentralt tema for API-et utover andre API-et matrikkelen tilbyr. Til slutt kommer to kapitler med tekniske detaljer og en opplisting av en del av tjenestene og metodene API-et tilbyr.

Tilbakemelding

Dokumentet er levende og endringsønsker, utvidelser og annet tas imot med glede. Vi oppfordrer alle til å lage JIRA-saker på matrikkelens JIRA med tilbakemeldinger.

1. Komme i gang

Nedlasting av wsdl-er

På nettsidene til hvert enkelt matrikkelsystem ligger det nedlastningslenker til wsdl-er og xsd-er for matrikkelAPI-et. I tillegg ligger det lenker direkte til wsdl-ene hvis man ønsker å hente ut enkelte av disse og laste ned direkte.

Bygging av klienter fra wsdl-er

Forskjellige utviklingsmiljø gjør dette på forskjellige måter. Det ligger her eksempler på hvordan man kan gjøre dette når man bruker Java og JAXB og når man bruker .NET og dingsen der.

Eksempel 1: wsimport for Java

Man kan benytte seg av verktøyet `wsimport` som følger med JDK (Java development kit) hvis man bruker Java som utviklingsverktøy på klientsiden. Man vil da kjøre følgende type kommando for hver enkelt wsdl man ønsker å benytte:

```
c:\temp\wstest>wsimport c:\temp\wstest\wsdls\AdresseServiceWS.wsdl -d
c:\temp\wstest\src -Xnocompile -keep -wsdllocation /wsdls/AdresseServiceWS.wsdl
```

I dette eksemplet ligger wsdl-ene i en katalog under `c:\temp\wstest\` kalt `wsdls`. Kommandoen vil generere nødvendige klasser for å kjøre kall mot `AdresseServiceWS`. Det er satt en del flagg på kommandoen og disse gjør følgende:

- -Xnocompile gjør at koden ikke blir kompilert men er klar for bruk inn i et annet program
- -keep gjør at man beholder kildefilene etter kjøring
- -wsdllocation gjør at man i generert kode får bestemt hvor `wsdl`-ene skal ligge relativt til klassene i programmet som bruker dem.
- -d angir hvor resultatet av kommandoen skal plasseres.

> Det er funnet noen rariteter ved kjøring av kommandoene. Hvis man skal ha inne flere `wsdl`-er i samme kildekodekataloger er det viktig at man kjører `StoreServiceWS` eller `RapportServiceWS` som siste `wsimport`-kommandoer. Hvis disse ikke er de siste kan man få problemer med `package-info.java` og kvalifisering av pakkenavn. Dette kan igjen skape problemer under kjøring.

Eksempel 2: .NET

Man kan benytte seg av verktøyet `svcutil` som følger med Visual Studio hvis man bruker dette utviklingsmiljøet. Det er også mulig å installere programmet ved å laste ned Microsoft sine SDK-er direkte, men det blir vanskelig å bruke den genererte koden til noe uten Visual Studio og bibliotekene den tilbyr. Verktøyet krever en mapping fra namespace til pakkenavn for alle pakker som skal være med i den genererte koden. Under følger en slik mapping for mange av pakkene man trenger å ha med for å generere kode for matrikkelen.

```
setlocal
set
nsmap_domain=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain,no.stat
kart.matrikkel.matrikkelapi.wsapi.v1.domain
set
nsmap_domain_adresse=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain
```

```
/adresse,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.adresse
set
nsmap_domain_adresse_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/
domain/adresse/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.adresse.k
oder
set
nsmap_domain_aktivitetsliste=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v
1/domain/aktivitetsliste,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.aktiv
itetsliste
set
nsmap_domain_aktivitetsliste_koder=/n:http://matrikkel.statkart.no/matrikkelapi/w
sapi/v1/domain/aktivitetsliste/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.
domain.aktivitetsliste.koder
set
nsmap_domain_bruker=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain/
bruker,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.bruker
set
nsmap_domain_bruker_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/d
omain/bruker/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.bruker.kode
r
set
nsmap_domain_bygning=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain
/bygning,no.statkart.matrikkel.matrikkelapi.wsapi.vbygning.domain.bygning
set
nsmap_domain_bygning_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/
domain/bygning/koder,no.statkart.matrikkel.matrikkelapi.wsapi.vbygning.domain.byg
ning.koder
set
nsmap_domain_endringslogg=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/d
omain/endringslogg,no.statkart.matrikkel.matrikkelapi.wsapi.vendringslogg.domain.
endringslogg
set
nsmap_domain_endringslogg_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsap
i/v1/domain/endringslogg/koder,no.statkart.matrikkel.matrikkelapi.wsapi.vendrings
logg.domain.endringslogg.koder
set
nsmap_domain_forretning=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/dom
ain/forretning,no.statkart.matrikkel.matrikkelapi.wsapi.vforretning.domain.forret
ning
set
nsmap_domain_geometri=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domai
n/geometri,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.geometri
set
nsmap_domain_geometri_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1
/domain/geometri/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.geometr
i.koder
set
nsmap_domain_grunnforurensing=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/
v1/domain/grunnforurensing,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.gru
nnforurensing
set
nsmap_domain_grunnforurensing_koder=/n:http://matrikkel.statkart.no/matrikkelapi/
wsapi/v1/domain/grunnforurensing/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v
1.domain.grunnforurensing.koder
set
nsmap_domain_jobb=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain/jo
bb,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.jobb
set
nsmap_domain_jobb_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/dom
ain/jobb/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.jobb.koder
set
nsmap_domain_kodeliste=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/doma
in/kodeliste,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.kodeliste
```

```
set
nsmap_domain_kommune=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain
/kommune,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.kommune
set
nsmap_domain_kommunetillegg=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1
/domain/kommunetillegg,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.kommune
tillegg
set
nsmap_domain_kommunetillegg_koder=/n:http://matrikkel.statkart.no/matrikkelapi/ws
api/v1/domain/kommunetillegg/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.do
main.kommunetillegg.koder
set
nsmap_domain_konsesjon=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/doma
in/konsesjon,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.konsesjon
set
nsmap_domain_konsesjon_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v
1/domain/konsesjon/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.konse
sjon.koder
set
nsmap_domain_kulturminne=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/do
main/kulturminne,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.kulturminne
set
nsmap_domain_kulturminne_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi
/v1/domain/kulturminne/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.k
ulturminne.koder
set
nsmap_domain_matrikkelenhet=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1
/domain/matrikkelenhet,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.matrikk
elenhet
set
nsmap_domain_matrikkelenhet_koder=/n:http://matrikkel.statkart.no/matrikkelapi/ws
api/v1/domain/matrikkelenhet/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.do
main.matrikkelenhet.koder
set
nsmap_domain_person=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain/
person,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.person
set
nsmap_domain_person_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/d
omain/person/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.person.kode
r
set
nsmap_domain_rapport=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain
/rapport,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.rapport
set
nsmap_domain_rapport_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/
domain/rapport/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.rapport.k
oder
set
nsmap_domain_sefrak=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain/
sefrak,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.sefrak
set
nsmap_domain_sefrak_koder=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/d
omain/sefrak/koder,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.sefrak.kode
r
set
nsmap_domain_util=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/domain/ut
il,no.statkart.matrikkel.matrikkelapi.wsapi.v1.domain.util

set
nsmap_service_adresse=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servi
ce/adresse,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.adresse
set
nsmap_service_aktivitetsliste=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/
```

```
v1/service/aktivitetsliste,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.ak
tivitetsliste
set
nsmap_service_bruker=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servic
e/bruker,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.bruker
set
nsmap_service_bygning=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servi
ce/bygning,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.bygning
set
nsmap_service_forretning=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/se
rvice/forretning,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.forretning
set
nsmap_service_grunnforurensing=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi
/v1/service/grunnforurensing,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.
grunnforurensing
set
nsmap_service_kodeliste=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/ser
vice/kodeliste,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.kodeliste
set
nsmap_service_kommune=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servi
ce/kommune,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.kommune
set
nsmap_service_konsesjon=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/ser
vice/konsesjon,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.konsesjon
set
nsmap_service_kulturminne=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/s
ervice/kulturminne,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.kulturminn
e
set
nsmap_service_matrikkelenhet=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v
1/service/matrikkelenhet,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.matr
ikkelenhet
set
nsmap_service_matrikkelsok=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/
service/matrikkelsok,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.matrikke
lsok
set
nsmap_service_nedlastning=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/s
ervice/nedlastning,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.nedlastnin
g
set
nsmap_service_person=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servic
e/person,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.person
set
nsmap_service_rapport=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servi
ce/rapport,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.rapport
set
nsmap_service_sefrak=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/servic
e/sefrak,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.sefrak
set
nsmap_service_store=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/service
/store,no.statkart.matrikkel.matrikkelapi.wsapi.v1.service.store
set
nsmap_exception=/n:http://matrikkel.statkart.no/matrikkelapi/wsapi/v1/exception,n
o.statkart.matrikkel.matrikkelapi.wsapi.v1.exception

set nsmap_domain_all=%nsmap_domain% %nsmap_domain_adresse%
%nsmap_domain_adresse_koder% %nsmap_domain_aktivitetsliste%
%nsmap_domain_aktivitetsliste_koder% %nsmap_domain_bruker%
%nsmap_domain_bruker_koder% %nsmap_domain_bygning% %nsmap_domain_bygning_koder%
%nsmap_domain_endringslogg% %nsmap_domain_endringslogg_koder%
%nsmap_domain_forretning% %nsmap_domain_geometri% %nsmap_domain_geometri_koder%
%nsmap_domain_grunnforurensing% %nsmap_domain_grunnforurensing_koder%
```

%nsmap_domain_jobb% %nsmap_domain_jobb_koder% %nsmap_domain_kodeliste%
%nsmap_domain_kommune% %nsmap_domain_kommunetillegg%
%nsmap_domain_kommunetillegg_koder% %nsmap_domain_konsesjon%
%nsmap_domain_konsesjon_koder% %nsmap_domain_kulturminne%
%nsmap_domain_kulturminne_koder% %nsmap_domain_matrikkelenhet%
%nsmap_domain_matrikkelenhet_koder% %nsmap_domain_person%
%nsmap_domain_person_koder% %nsmap_domain_rapport% %nsmap_domain_rapport_koder%
%nsmap_domain_sefrak% %nsmap_domain_sefrak_koder% %nsmap_domain_util%
%nsmap_service_adresse% %nsmap_service_aktivitetsliste% %nsmap_service_bruker%
%nsmap_service_bygning% %nsmap_service_forretning%
%nsmap_service_grunnforurensing% %nsmap_service_kodeliste%
%nsmap_service_kommune% %nsmap_service_konsesjon% %nsmap_service_kulturminne%
%nsmap_service_matrikkelenhet% %nsmap_service_matrikkelsok%
%nsmap_service_nedlastning% %nsmap_service_person% %nsmap_service_rapport%
%nsmap_service_sefrak% %nsmap_service_store%

```
svcutil /out:WebServiceProxy.cs /serializer:DataContractSerializer
/config:App.config %nsmap_domain_all%  wsdls\*.xsd wsdls\*.wsdl
```

Denne kjøringen krever at man har lastet ned zip-filen i en underkatalog med navn `wsdls`. Hvis man ikke erstatter teksten "REPLACE_WITH_ACTUAL_URL" i port-elementene i wsdl-ene med en ekste URL vil man få feilmeldinger under genereringen på dette og man vil mangle transport-egenskapen i app.config, men den genererte koden blir ok. Det kan derfor være lurt å legge inn en URL i wsdl-filene under generering og så skille dette ut i en property senere for å kunne konfigurere dette som man vil.

Kjøring av kall mot API-et

Etter at koden er generert for webtjenesteklienten blir det neste steget å kalle på tjenestene. Eksempel på dette for Java følger

Eksempel

```
AdresseService service = new AdresseServiceWS().getAdresseServicePort();
//Oppretter instansen vår

BindingProvider provider = (BindingProvider)service;
provider.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
endpointURL ); //Bytt ut med ekte URL
provider.getRequestContext().put(BindingProvider.USERNAME_PROPERTY, username);
//Bytt ut med faktisk brukernavn
provider.getRequestContext().put(BindingProvider.PASSWORD_PROPERTY, password);
//Bytt ut med faktisk passord

VegadresseId id = service.findAdresseIdForIdent(new VegadresseIdent(), new
MatrikkelContext());
```

Koden over gir oss tilgang til en tjeneste, setter URL, brukernavn og passord på tjenesten og utfører så et enkelt kall (med ikke-fullstendige parametre).

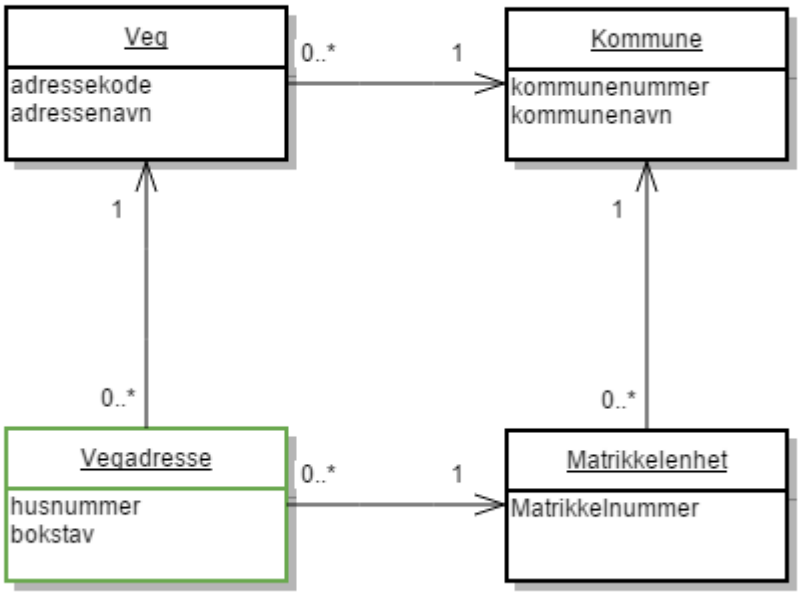
2. Boblemodellen

Hva er bobler?

Matrikkelens domenemodell er bygget opp basert på det logiske domenet modellen representerer. Implementasjonsmodellen er delt inn i selvstendige objekter som refererer til hverandre. Det som gjør denne modellen spesiell er at objektene er løskoblet ved at de refererer til hverandre via koblinger på id-nivå og ikke på objekt-nivå, som i en relasjonsdatabase. Dette fører til at vi kan se på et enkelt objekt alene uten å måtte laste inn andre objekter før vi ønsker å navigere til dem.

Eksempel:

Under følger en forenklet modell for vegadresse og koblingene den har mot andre objekter.



Det er her mange koblinger mellom objekter, men vi ser at alle disse objektene er selvstendige og kan da modelleres som egne bobler. I kode blir dette da følgende

```
class Kommune {
 KommuneId id;
 String kommunenummer;
 String kommunenavn;
}

class Veg {
 VegId id;
 int adressekode;
 String adressenavn;
 KommuneId kommuneId;
}

class Matrikkelnummer {
 KommuneId kommuneId;
 int gardsnummer;
 int bruksnummer;
 int festenummer;
 int seksjonsnummer;
}

class Matrikkelenhet {
 Matrikkelnummer matrikkelnummer;
}

class Vegadresse {
 VegadresseId id;
 int husnummer;
 char bokstav;
 VegId vegId;
 MatrikkelenhetId matrikkelenhetId;
}
```

Det vi ser her er at alle objekter har sin egen id som er av en type som representerer klassen. Disse id-klassene inneholder igjen et unikt tall som identifiserer objektet. Vi kan så bruke disse id-ene til å referere til de andre objektene i stedet for å vite om objektene i seg selv. En vegadresse vet da om id-en til vegen den er koblet til, men den vet ikke hva adressenavnet er for denne vegen. Det er det kun vegen selv som vet. På samme måte vet vegen hvilken id kommunen den ligger i har, men den vet ikke hva kommunen heter. Det er det kun kommunen som vet.

Bobler og komponenter

I eksemplet over er det en klasse, **Matrikkelnummer**, som ikke finnes som en egen boks i den logiske modellen. Dette er en klasse vi kun lager for å enkelt samle informasjon som henger sammen men som ikke kan sees på for seg som et selvstendig objekt. I dette tilfellet er matrikkelnummeret feltene som logisk identifiserer en matrikkelenhet, men det kan være andre klasser av samme type. Eksempler på dette er en etasje på et bygg, eller et eierforhold for en matrikkelenhet, klasser som har verdi, men som ikke er selvstendige. Disse objektene er en del av boblene de henger på og kan ligge direkte som felter i boblene, eller som del av en liste av komponenter på boblen. Disse objektene følger alltid med boblen de er en del av og kan ikke søkes opp utenom boblene.

Navigering blant bobler

Boblene refererer til hverandre, som vi så i eksempelet over, ved hjelp av id-er. Domenemodellen er ganske tett koblet på den måten at vi ofte vil ønske å sette sammen data fra forskjellige bobler i en visning eller databehandling. Eksempelvis kan vi tenke oss at vi ønsker å vise en adresse for en person. Man er vant til å se

adresser med et navn først etterfulgt av et husnummer og bokstav før man til slutt får postnummer og poststed. Denne informasjonen finnes i matrikkelen og i objektene våre, men vi må gå mellom flere objekter og hente data forskjellige steder for å sette sammen disse elementene. For å gjøre dette benytter vi oss av en tjeneste med navn **StoreService**. Denne tjenesten tilbyr metoder for å hente ut et eller flere objekter basert på deres id, og vi kan bruke dette for å hente ut de objektene vi ønsker å navigere til.

Eksempel

```
StoreService storeService; //Oppretter denne slik vi trenger i rammeverket vårt.

//Vi har en vegadresseId fra før. Vi ønsker å hente ut denne og navigere rundt i
modellen basert på den.
VegadresseId vegadresseId = new VegadresseId(12345L);

Vegadresse vegadresse = storeService.getObject(vegadresseId);
Veg veg = storeService.getObject(vegadresse.getVegId());

Kommune kommune = storeService.getObject(veg.getKommuneId());
```

Slik kan vi fortsette for å navigere oss gjennom modellen. Det blir fort tydelig at det kan bli mange kall til tjeneren over nettverket hvis man gjør det så enkelt, så vi anbefaler å benytte seg av caching av boblene på klientsiden for lettere bruk. Under følger et enkelt eksempel på en slik cache.

Eksempel 2: Cache

```
class StoreCache{
 Cache<MatrikkelBubbleId, MatrikkelBubbleObject> simpleCache;
 StoreService storeService;

 public MatrikkelBubbleObject get(MatrikkelBubbleId id){
  if(!simpleCace.containsKey(id)){
   MatrikkelBubbleObject o = storeService.getObject(id);
   simpleCache.put(o);
  }
  return simpleCache.get(id);
 }

 public Collection<MatrikkelBubbleObject> get(Collection<MatrikkelBubbleId> ids){
  Set<MatrikkelBubbleId> idsToGet = new HashSet<>();
  for(MatrikkelBubbleId id : ids){
    if(!simpleCache.containsKey(id){
     idsToGet.add(id);
  }
  simpleCache.putAll(storeService.getObjects(idsToGet));
  return simpleCache.getAll(ids);
 }
}
```

Dette er en veldig enkel implementasjon, og man må nok se på å la objekter forsvinne fra cachen etter en viss tid, eller tømme cachen med jevne mellomrom for å sikre seg at man alltid har de nyeste versjonene av data. Konseptet er uansett at man ikke trenger å hente en boble fra tjeneren hvis man har den fra før! Dette gjør navigasjonen mye raskere, og man kan gå mellom objekter uten anstrengelse.

3. Historikk

Historikk i matrikkelen

I "lov om eigedomsregistrering" (matrikkelloven) og dennes forskrifter beskrives krav og bestemmelser rundt føring og lagring av data i matrikkelen. I tillegg spesifiserer arkivloven en del krav rundt lagring av føringer og resultater av dette. For å tilfredsstille kravene og ønskene i lovverket har det i matrikkelen blitt implementert en arkivarisk historikk der alle føringer tas vare på i systemet. Dette kan da brukes for å se tidligere situasjoner. I dagens system benyttes dette i hovedsak i feilrettinger og ved klager på føring.

Historikken i matrikkelen skal tilfredsstille geodatalovens forskrifter når det gjelder implementasjon av INSPIRE direktivet. Av dette følger en del feltnavn som blir beskrevet her.

Innføring av historikk i matrikkelen og det nye matrikkelAPI-et har noen konsekvenser for modellen og tjenestene man benytter seg av. Dette dokumentet beskriver hvordan vi kan se historikk og hvordan man henter historiske data i tjenestene.

Historikk i modellen

Hver endring av et objekt i matrikkelen fører til at systemet tar vare på en ny versjon av objektet. På denne måten får vi en tidslinje av versjoner der det er mulig å ta steg bakover og se forskjeller fra versjon til versjon. Hver enkel versjon er komplett og inneholder alle data slik de var i tidsrommet versjonen var gyldig. For å synliggjøre dette og gjøre det mulig å verifisere dette er det lagt til nye felter i systemet. Disse feltene er:

- oppdateringsdato
- oppdatertAv
- sluttdato
- avsluttetAv
- versjonId

Disse feltene beskriver en versjon av et objekt og er realiserte på både bobler og komponenter i matrikkelen. Ikke alle bobler er av typer som har historikk, men de fleste domeneboblene har dette.

Feltene `oppdateringsdato` og `oppdatertAv` viser når boblen sist ble endret, og hvilken matrikkelbruker som utførte endringen. Disse feltene og `versjonId` er alltid satt på en boble som har disse feltene. `VersjonId` er et felt som teller opp hver gang man endrer et objekt. Man kan dermed se på en boble eller en komponent hvor mange ganger brukere har gjort noe med dem.

Feltene `sluttdato` og `avsluttetAv` er kun satt til ekte verdier på versjoner av objekter som er historiske. Dette kan være fordi noen har oppdatert objektet og vi har fått en ny versjon, eller det kan være fordi noen har slettet objektet. Hvis det er snakk om en oppdatering vil man ha en ny versjon som har `oppdateringsdato` lik denne versjonens `sluttdato` og `versjonId` lik dennes `versjonId` + 1.

En versjons levetid er definert som fra inklusivt `oppdateringsdato` til eksklusiv `sluttdato`.

> **sluttdato for sanntidsversjoner**
>
> Hvis et objekt ikke har noen sluttdato satt er datoen satt til "nåtid". Dette er en kunstig dato langt inn i framtiden som symboliserer at dette er et levende objekt. Dette er det samme som vi benytter i `matrikkelContext` som blir sendt inn i alle webtjenestekall for å vise at vi ønsker sanntidsdata.
>
> Denne datoen er: `9999-01-01 00:00:00.00`

Eksempel:

Vi har en adresse som kun har én versjon. Denne oppstod da adressen ble opprettet. Den har følgende data:

```
Vegadresse
 id = 2
 husnummer = 10
 vegId = 1
 oppdateringsdato = 01.01.2016 kl 00:00:00.00
 oppdatertAv = bruker1
 versjonId = 1
 sluttdato = 01.01.9999 kl 00:00:00.00
 avsluttetAv = null
```

Vi endrer så husnummer på adressen. I systemet vil det da eksistere to versjoner av objektet som ser slik ut:

```
Vegadresse
 id = 2
 husnummer = 10
 vegId = 1
 oppdateringsdato = 01.01.2016 kl 00:00:00.00
 oppdatertAv = bruker1
 versjonId = 1
 sluttdato = 12.01.2017 kl 11:35:21.000123
 avsluttetAv = bruker2

og

Vegadresse
 id = 2
 husnummer = 999
 vegId = 1
 oppdateringsdato = 12.01.2017 kl 11:35:21.000123
 oppdatertAv = bruker2
 versjonId = 2
 sluttdato = 01.01.9999 kl 00:00:00.00
 avsluttetAv = null
```

Det er nå objektet med `versjonId` 2 som er den aktive versjonen av objektet, mens versjon 1 har blitt historisk.

Navigering og historikk

Det er kun objektene som blir endret som får en ny versjon ved endring, og ikke andre objekter. Det betyr at man ender opp i situasjoner der navigeringen av objektgrafen vil gi mange forskjellige oppdateringsdatoer på objekter som hører sammen. Man må da benytte seg av `oppdateringsdato` for én boble for å finne ut hvilken versjon av boblene den peker på vi skal ha tak i. For sanntidsnavigering blir dette enklere da vi alltid er interessert i den levende versjonen av bobler, men for historikk kan dette bli mer komplisert.

Det vil alltid kun være én versjon av en boble som gjelder for et gitt tidspunkt så hvis man har et startpunkt for navigering der man vet hvilket tidspunkt man vil se på vil systemet alltid gi ut de riktige versjonene av bobler som gjelder for det angitte tidspunktet.

Uthenting av historikk

Som skrevet i infoboksen over angir vi i `matrikkelContext`-parameteren (se "Tekniske detaljer" for mer informasjon om utfylling av denne) til alle tjenester på matrikkelAPI-et tidspunktet vi ønsker å se på data for. I de fleste tilfeller ønsker vi den levende versjonen og vi angir da følgende:

```
matrikkelContext
 ...
 snapshotVersion = 01.01.9999 00:00:00.00
 ...
```

Dette viser for systemet at man ønsker sanntidsdata. Hvis man i stedet putter inn en historisk dato her vil man gjøre søk og hente ut historiske data. Dette gjelder for de fleste tjenestene, inkludert `StoreService` som brukes for å hente objekter.

> For å kunne hente ut historiske data kreves en spesiell rolle på brukeren. Denne rollen tildeles av Kartverket men det er foreløpig svært begrenset hvem som kan få denne rollen.

StoreService.getVersions(...)

For å se hvilke oppdateringsdatoer som eksisterer for en boble kan man benytte seg av metodene `getVersions` og `getVersionsForList` på `StoreService`. Disse metodene gir ut informasjon tidspunkt for alle oppdateringer gjort på en gitt boble innenfor et tidsrom.

Eksempel:

```
idsMedVersjon = storeService.getVersions(
 new KommuneId(233),
 new Timestamp("01.01.2012"),
 new Timestamp("01.01.2017")
);

idsMedVersjon er da:
(
 ("27.10.2013T01:00:00.000+02:00", 233),
 ("16.02.2016T18:33:58.113+02:00", 233)
)
```

Vi ser at returen er en samling av nøkkel-verdi par der nøkkelen er tidspunktet kommunen ble oppdatert på og verdien er id-verdien.

Modellendringer

Over tid vil domenemodellen til matrikkelen endres, men for å sikre at de historiske versjonene fortsatt kan gjengis som de var på tidspunktet de ble opprettet er det lagt noen bånd på endringer som kan gjøres i modellen. Det er også innført et felt på alle bobler for å vise hvilke felt som inneholder data for denne versjonen av objektet.

Lovlige endringer

- Tillegg av felter
- Fjerning av felter (for framtidige data, ikke for gamle data)

Ulovlige endringer

- Endring av type for eksisterende felt
- Navneendring på felt

Modellering av endringer

Alle bobler har i API-et et eget felt, `metadata`, som inneholder hvilke feltnavn objektet har data i. Dette betyr at man kan se på dette for en enkelt versjon av en boble og se med en gang om det at et felt inneholder `null` er fordi dette er en gyldig verdi, eller fordi feltet ikke eksisterte på tidspunktet versjonen oppstod.

Det står over at fjerning av et felt er en lovlig endring, og det er sant hvis man ser på metadataene. Feltet vil ikke forsvinne fra modellen, men det vil ikke lengre være mulig å fylle data inn i. Sanntidsversjoner av objekter vil da ikke ha data i feltet, men historiske versjoner vil fortsatt kunne ha data der.

4. Tekniske detaljer

MatrikkelContext

Alle tjenestekall mot API-et har en MatrikkelContext som siste parameter. Denne setter opp en del innstillinger for kallet og dataene som skal returneres. Blant disse er versjon av systemet klienten er laget for, identifikasjon av klienten, koordinatsystem man ønsker data returnert i og hvilket tidspunkt man ønsker data for. Under følger eksempel og litt mer detaljer for noen av feltene.

Eksempel

```
MatrikkelContext context = new MatrikkelContext();
context.setLocale("no_NO_B"); //Norsk bokmål
context.setBrukOriginaleKoordinater(false);
context.setKoordinatsystemKodeId(new KoordinatsystemKodeId(10L)); //Denne bør
hentes via kodeliste for ønsket koordinatsystem
context.setKlientIdentifikasjon("minKlient");
context.setSystemVersion("3.10"); //Det er denne versjonen jeg har hentet ut
wsdl-er for og bygget min lokale kode for.
context.setSnapshotVersion(new Timestamp("9999-01-01T00:00:00+01:00")); //Siste
versjon av objekter
```

Klientidentifikasjon

En streng som identifiserer klienten. For noen kall er det påkrevd med en godkjent klient, men for de fleste innsynskall må denne bare være utfylt med noe.

SystemVersion

Hvilken versjon av matrikkelen man har laget klienten basert på. Ved å angi denne sørger API-et for at man ikke får ut felter og datainnhold man ikke kjenner til. API-et er bakoverkompatibelt og sørger for å mappe inn nye subtyper av objekter i kjente supertyper slik at man får ut alt man kan.

SnapshotVersion

Angir om man ønsker å gå mot sanntidsversjonen av matrikkeldata eller en historisk versjon. For systemer som har historikk er det bestemt at sanntids-SnapshotVersion skal være 01-01-9999 klokken 00:00:00. Alle andre tidspunkt enn dette vil føre til at man prøver å hente ut historiske data, og dette krever en egen rolle for brukeren som skal gjøre kallet. Det er derfor viktig å passe på at man skriver tidspunktet som beskrevet i eksemplet over.

Koder

En del felter i modellen har en fast liste av mulige verdier, en kodeliste. Det er definert opp et sett av regler for koder, bruk av disse og måten vi behandler dem.

1. Id for en kode skal aldri gjenbrukes
2. Kodeverdi for en kode skal aldri gjenbrukes
3. Beskrivelse for en kode kan endres, men ikke slik at betydningen endres, bare skrivefeil og lignende.
4. Koder kan oppstå og forsvinne når som helst. Det er derfor viktig å synkronisere lokale koder mot systemet ofte.

Kodene i matrikkelen er bygget opp av id, kodeverdi, beskrivelse og ekstra felter. Id-ene er unike innenfor kodelisten, men ikke unike for hele systemet. Kodeverdien og beskrivelsen er ofte bestemt av en standard og

beskrivelsen er lokalisert.

Det er laget en egen tjeneste for synkronisering av kodelister, **KodelisteService**. Denne har en metode for å hente ut alle kodelister.

Forskjellige typer bobler

Det finnes flere forskjellige typer bobler i matrikkelsystemet. Noen har ekstra felter knyttet til seg, mens andre er enklere. Her følger en beskrivelse av de forskjellige typene.

MatrikkelBubbleObject

Grunnobjektet for alle bobler. Alle bobler arver fra denne klassen. Ved å arve fra denne klassen viser objektene at de er bobler, at de har en unik numerisk id som identifiserer dem, og at de har et felt av **metadata** som viser hvilke felter på objektet som inneholder informasjon.

Kode

Se **koder** over.

MatrikkelBubbleObjectWithHistory

De fleste boblene er av denne typen. Typen tilbyr følgende felter:

- oppdateringsdato
- sluttdato
- versjonId
- oppdatertAv
- avsluttetAv
- versjon

Dette er versjoneringsfelter som beskriver når et objekt har blitt endret, hvem som har endret det og hvor mange ganger objektet har blitt endret. For sanntidsversjoner av objekter vil ikke `sluttdato` eller `avsluttetAv` ha noen verdi, men for historiske versjoner av objekter vil disse være fylt inn for å vise hvem som avsluttet en versjon av objektet. `Sluttdato` for ett objekt vil være `oppdateringsdato` for den neste versjonen av objektet hvis dette har blitt oppdatert.

5. Tjenester

Tjenesteoppsett

Tjenestene i matrikkelAPI-et er delt opp basert på type, med noen spesialtilfeller som unntak. Dette betyr at søk der vi ønsker å hente ut adresser ligger i AdresseService, søk der vi ønsker å hente ut matrikkelenheter ligger i MatrikkelenhetService og så videre.

Unntak er spesialtjenester som StoreService for å hente ut objekter, KodelisteService for å hente ut koder, RapportService for bestilling og uthenting av rapporter og NedlastningService for å laste ned store mengder data.

De alle fleste tjenestemetodene returnerer id-er for det man har søkt på. Man skal så bruke StoreService for å hente ut objekter.

Typer tjenestemetoder

Tjenestemetodene på de vanlige tjenesteinterfacene kan deles inn i tre typer.

- Søk basert på en sammensatt modell
- Søk for å finne id for en ident
- Inversrelasjonssøk

Beskrivelse av de tre typene følger

Søk basert på sammensatt modell

Noen søk kan basere seg på input fra mange forskjellige deler av domenet. Det er for disse laget egne tjenestemoder som tar inn søkemodeller som parameter. Eksempel på dette er **MatrikkelenhetService** sin metode **findMatrikkelenheter**. Denne metoden tar inn en klasse med navn **MatrikkelenhetsokModel** som parameter og denne klassen gir oss mulighet til å spesifisere alt fra matrikkelenhetfelter som gårdsnummer og bruksnummer, til koblingsfelter som krav om jordskifte og over til koblede objekter slik at vi kan angi adressenavn og husnummer for en adresse tilknyttet matrikkelenheten, eller etternavnet til en person som har eierforhold på matrikkelenheten. Siden tjenesten ligger på **MatrikkelenhetService** vet vi at det er **MatrikkelenhetId**-er som blir returnert fra søket.

Det er laget slike sammensatte søk for adresser, bygg, matrikkelenheter og person i API-et.

Søk for å finne id for en ident

Vi kan ikke forvente at brukere av API-et vet hva våre unike id-er er for de forskjellige objektene våre så det er laget tjenester som tar inn en logisk identifikator for et objekt og returnerer id-objektet som representerer denne i systemet. Disse tjenestene finnes for alle boblene som har logiske identifikatorer, men ikke for absolutt alle bobler. De boblene som ikke har noen logisk identifikator (som for eksempel en teig, en flate for en matrikkelenhet) vil ikke ha tjenester for å oversette fra logiske identifikatorer til id-er.

Det er laget både entalls og flertalls-versjoner av tjenestemetodene slik at man kan hente ut id-er for mange identer om gangen hvis man ønsker det.

Klasser som har disse tjenestene er eksempelvis Adresse, Veg, Bygg, Forretning, Kommune og Matrikkelenhet.

Inversrelasjonssøk

I modellen kan man alltid navigere via id-koblingene som er laget, men det kan også være tilfeller der man ønsker å gå motsatt vei. Det er i disse tilfellene laget tjenester for å søke opp dette. Eksempelvis kan man i **AdresseService** søke opp alle adresser for en eller flere veger eller alle adresser for en eller flere bygg. Disse

koblingene finnes ikke direkte i modellen nødvendigvis men det er mulig å navigere i modellen for å finne objektene.

Spesielle tjenester

StoreService

Denne tjenesten tilbyr metoder for å hente ut en eller flere bobler basert på deres id. Denne er primærmåten å navigere gjennom boblemodellen og for å hente ut boblene.

RapportService

Matrikkelen tilbyr en del predefinerte rapporter og denne tjenesten brukes for å bestille og hente disse. Bruksmønsteret er at man benytter en metode for å bestille en rapport. Man får da tilbake et rapport-objekt som peker til et Jobb-objekt som man kan hente ut for å få status for bestillingen. Når rapporten er ferdig kan man hente denne ut over HTTPS som en filnedlasting.

Eksempel

Følgende eksempel viser hvordan vi kan bestille og hente ut en rapport. Bruker rapporten "Samlet rapport for matrikkelenhet" som eksempel.

```
//Finner først matrikkelenheten vi vil ha rapport for
MatrikkelenhetService matrikkelenhetService;
MatrikkelenhetId matrikkelenhetId =
matrikkelenhetService.findMatrikkelenhetIdForIdent(new MatrikkelenhetIdent(...));

//Bestiller så rapporten
RapportService rapportService;
OfflineMatrikkelRapport r =
rapportService.createSamletRapportForMatrikkelenhet(matrikkelenhetId,
ExportTypeId.PDF, true)

//Henter rapporten ut og finner URL for filen.
MatrikkelRapport rapport = rapportService.hentRapport(r.getJobbId()); //Dette
kallet feiler hvis rapporten ikke er klar.

String urlForRapport = rapport.getURL(); //Kan så bruke denne for å hente
rapporten over https
```

KodelisteService

API-et tilbyr en tjeneste for å hente ut alt av koder i systemet. Dette kan da brukes for å cache opp kodeverdier i lokal klient. Da det i utgangspunktet kan komme til nye kodeverdier når som helst bør man i klient benytte seg av denne tjenesten ofte for å være sikker på at man ikke ender opp med å få bobler fra StoreService som refererer til kodeverdier man ikke vet om.

Metoden getKodelister på tjenesten tar inn et tidspunkt, men dette bør ikke benyttes med mindre man vet at man ikke ønsker kodelister på nåværende tidspunkt. Dette krever spesielle rettigheter så man bør nok benytte seg av "nåtid" i de fleste tilfeller.

NedlastningService

Det kan være tilfeller der man ønsker å hente ut store datamengder. NedlastningService tilbyr to metoder for å

kunne gjøre dette. De to metodene har samme signatur men forskjellige returtyper. Den ene metoden gir tilbake id-er og den andre gir oss boblene. Det er opp til brukeren å bestemme hvilke av disse som er mest hensiktsmessige. Hvis man allerede har en lokal kopi av matrikkelen og vil sjekke om man har alle id-er vil det være unødvendig å hente ut alle objekter. Under følger eksempel på bruk av tjenesten

Eksempel

```
NedlastningService service;

String filter = "{kommunefilter: [\"1201\"]}";
List<VegId> alleIds = new ArrayList<>();
List<VegId> idsEtterId;
VegId sisteId = null;
do {
 idsEtterId = service.findIdsEtterId(sisteId, Veg.class, filter, 10000);
 alleIds.addAll(idsEtterId);
 sisteId = alleIds.get(alleIds.size() - 1);
} while(idsEtterId.size() > 0);

return alleIds;
```

Eksemplet henter ut alle veger i kommune med kommunenummer "1201". Se beskrivelsen av tjenesten for mer informasjon rundt de forskjellige parametrene.

Tjenesten kan man da bruke for å hente ut data for en eller flere kommuner (eller hele landet) og man kan så koble dette sammen med endringsloggstjenester (foreløpig kun i gammelt v3 endringsloggAPI) for å bygge opp en lokal kopi av matrikkeldata.

EndringsloggService

EndringsloggService kan brukes for å hente en instans av `Endringer`, som igjen inneholder en liste av `Endring`. En `Endring` representerer en Nyoppretting, Oppdatering, Sletting eller en Typeendring for ett bobleobjekt. Legg merke til at en endring inneholder ikke hva som eventuelt har blitt endret i bobleobjektet, om man oppgir ReturnerBobler.Alltid som parameter til endringsloggstjenestenvil man få tilbake boblen i nåværende tillstand(dvs. ikke tilstanden boblen hadde da endringen ble utført).

Brukstilfelle for denne tjenesten kan være å holde ett system synkronisert med matrikkelen. Da bruker man gjerne nedlastingstjenesten først, og bruker da den siste endringsid-en man fikk fra denne som første endrings id til endringsloggstjenesten.

Eksempel som går igjennom alle endringer

```
EndringsloggService endringsloggService;
StoreService storeService;

Map<BubbleId<?>, BubbleObject> boblerForHenting = new HashMap<>();
Set<BubbleId<?>> leggesTil = new LinkedHashSet<>();
Set<BubbleId<?>> oppdateres = new LinkedHashSet<>();
Set<BubbleId<?>> slettes = new LinkedHashSet<>();

String filter = "{kommunefilter: [\"1201\"]}";
MatrikkelEndringId forsteEndringsId = null; // bruk f.eks. id fra siste endring
hentet med nedlastingstjenesten
Endringer<MatrikkelEndring<?, ?>> endringer;
do {
   endringer = endringsloggService.findEndringer(
         forsteEndringsId,             // returner endringsobjekter fra og med
oppgitt endringsID.
         MatrikkelBubbleObject.class,  // returner endringer for alle typer i
matrikkelen
         filter,                       // begrens til endringer for en gitt
kommune
         ReturnerBobler.Aldri,         // ikke returner bobleobjektene endringen
gjelder for(dvs. kun endringsobjektet)
         10000);                       // returner maksimalt 10000 endring

   // Samle opp id for alle bobleobjekter som er lagt til eller oppdatert, slik
at de kan hentes i ett store service kall
   for (MatrikkelEndring<?, ?> endring : endringer.getEndringList()) {
      switch (endring.getEndringstype()) {
         case Nyoppretting:
            leggesTil.add(endring.getEndretBubbleId());
            boblerForHenting.put(endring.getEndretBubbleId(), null);
            break;
         case Typeendring:
         case Oppdatering:
            oppdateres.add(endring.getEndretBubbleId());
            boblerForHenting.put(endring.getEndretBubbleId(), null);
            break;
         case Sletting:
            slettes.add(endring.getEndretBubbleId());
            break;
      }
   }
} while (!endringer.isAlleEndringerFunnet());

// hent alle nye og oppdaterte bobler med store service
for (BubbleObject bubbleObject :
storeService.getObjects(boblerForHenting.keySet())) {
   boblerForHenting.put(bubbleObject.getId(), bubbleObject);
}

for (BubbleId<?> bubbleId : leggesTil) {
   BubbleObject bubbleObject = boblerForHenting.get(bubbleId);
   // legg til bubbleObject til system som skal synkroniseres
}

for (BubbleId<?> bubbleId : oppdateres) {
   BubbleObject bubbleObject = boblerForHenting.get(bubbleId);
   // oppdater bubbleObject i system som skal synkroniseres
}

for (BubbleId<?> bubbleId : slettes) {
   // slett bubbleId fra system som skal synkroniseres
}
```