# DBH-API koding

## Eksempel på kode og programbibliotek for å bruke DBH-API

Noen enkle eksempler på kode for å komme i gang med å bruke API-et.

# Innhold



## Veiledning for spørringer mot API

Data kan hentes via Curl eller fra andre program.

Data er organisasjonsdata, studietilbud, studentdata, doktorgradsdata, økonomidata, publiseringsdata fra kanalregisteret, personaldata og andre data(som for eksempel areal) . Dette utgjør en statistikkbank.

Spørringer mot DBH API er p.t tilgjengelig via Http REST - POST og JSON format (JavaScript Object Notation). Du kan endre inputverdiene ('values') programmatisk. POST er valgt fordi det gir en mer fleksibel måte å spørre på i forhold til GET. Vanligvis angir POST oppdatering. Dette er ikke tilfelle i API spørringen.

Logon fra maskin software via Json Web Token (JWT) vil vare i 1 time. Bruker må selv lage rutiner for å hente nytt token før det gamle er gått ut på tid. For eksempel i form av en loop. Der er eksempel på dette i dette dokumentet.

Data som genereres via API blir ikke lagret hos DBH.

Det er flere veier inn til DBH data via API.

1. Via klient – https://api.nsd.no/dbhapiklient/.
2. Programvare. Bruk av 'cUrl', 'Post', 'R', Javascript eller annen programvare.

Bruk av API via programvare krever pålogging via klient_id og secret_id.

Vi tenker oss at når bruker skal sette opp en maskin til maskin kobling, så går han først inn i API klienten og prøver ut spørringen. Deretter kan han prøve spørringen i cUrl og til sist kan han programmere inn i egne program.  cUrl finnes på de fleste Linux maskiner.

Hvis spørringen blir lastet ned på Windows maskin og deretter overført til Linux, må tegnsettet konverteres (f.eks Dos2Unix ev. Unix2Dos). Vi ser feil ved tegnsett hvis æøå er feil. For eksempel Å...rstall.  Linjeskift er også forskjellig fra dos til unix format.

Systemet er et 'plug and play' system og det er laget en 'toolboks'(ligger på DBH sine nettsider) med eksempel på  queries mot de fleste tabellene.

Noen av variablene, for eksempel 'Institusjonskode' eller 'Avdelingskode', er NSD sine egne verdier og er slik sett ikke offisielle verdier.

## Statuskode

Sammen med http metoder som API responderer til, vil det bli returnert http statuskoder.

Status 200 indikerer at request er ok.

Returkoder i 400 segmentet, indikerer at noe gikk galt. Det kan være syntaks i JSON, Innhold i JSON eller at du prøver å gjøre noe du ikke er autorisert for. Dette er klientside problem og skyldes gjerne feil ved JSON inn eller innholdet i denne.

Returkoder i 500 segmentet indikerer et server-side problem, slik at forespørselen ikke kan utføres. Du får det også når du ikke får kontakt med server.

## Teknisk informasjon for å hente data fra API via program - cUrl

Illustrert med cUrl.
Det forventes at bruker behersker cUrl.

Returdata er JSON eller CSV, avhengig av om du velger å kalle /hentCSVTabellData eller /hentJSONTabellData. For det siste valget er det mulig å få ut en statuslinje.

URL som du skal bruke, vil  stå i konfigureringsbildet for API'et. Den brukes i forbindelse med https 'POST' spørring over linje til NSD.

Hent token:
curl --data "grant_type=client_credentials" --user sso_id:sso_secret https://sso.nsd.no/oauth/token

Klient-id og secret-id må institusjonen selv administrere via NSD sitt brukersystem når dette er klart. Foreløpig må institusjonen gi beskjed til NSD om å legge inn klientmaskinen.

Eksempel på cUrl spørring av rapport 060.:
export JWT=<sett inn token som er hentet her>
curl -o ./Result/060-result.dat -i -d "@./queryBody/060-querybody.json" -H "Content-Type: application/json;charset=UTF-8" --header "Authorization: Bearer $JWT" -v -X POST https://api.nsd.no/dbhapitjener/Tabeller/hentJSONTabellData 2>&1
Her blir query hentet fra ./queryBody/060-querybody.json og data lagt i /Resultat/060-resultat.dat
Etter først å ha fått et gyldig token (pålogging) fra NSD.

```
:set fileformat=unix
#!/bin/bash
# -i gir informasjon om overføring av data.
echo "Starter uthenting av fil 060."
export JWT=
curl -o ./Result/060-result.dat -i -d "@./queryBody/060-querybody.json" -H "Content-Type: application/json;charset=UTF-8" --header "Authorization: Bearer $JWT" -v -X POST https://api.nsd.no/dbhapitjener/Tabeller/hentJSONTabellData 2>&1
```

-i gir statusinformasjon og kan tas bort etter ønske.

## Eksempel på AJAX query fra klient

(Representert ved javascript biblioteket AXIOS).

stitilapi = <sti til api>
jsonquery = innhold i spørring boksen i bildet.

```
  let config = {
    onUploadProgress: function(progressEvent) {
      let percentCompleted = Math.round( (progressEvent.loaded * 100) / progressEvent.total );
    }, //progressindikator
    headers: {
      'Authorization': 'Bearer ' +  this.props.hashToken,
   }
  };
```

Spørring via Javascript - Axios:

```
  Axios.post(stitilapi, {
    node: jsonquery
  },config)
 .catch(error => {
      if (error.response) {
          .....
       } else if (error.request) {
          .....
       } else {
          .....
      }
 });
```

HTTP error vil vise i linjen for Statuskode i konsollet eller leses ut fra "error" seksjonen.
I Axios er ikke \ nødvendig, slik som det er gjort i eksempelet nedenfor.

## cUrl: Eksempel på spørringen med en inputfil: querybody.json

Denne filen kopieres direkte fra spørringen som er lastet ned fra klienten .

Om det blir brukt Linux kan filen kopieres fra Windows til Linux. På Linux må det kjøres 'fromdos' kommando, eller tilsvarende, for å konvertere tegnsett.

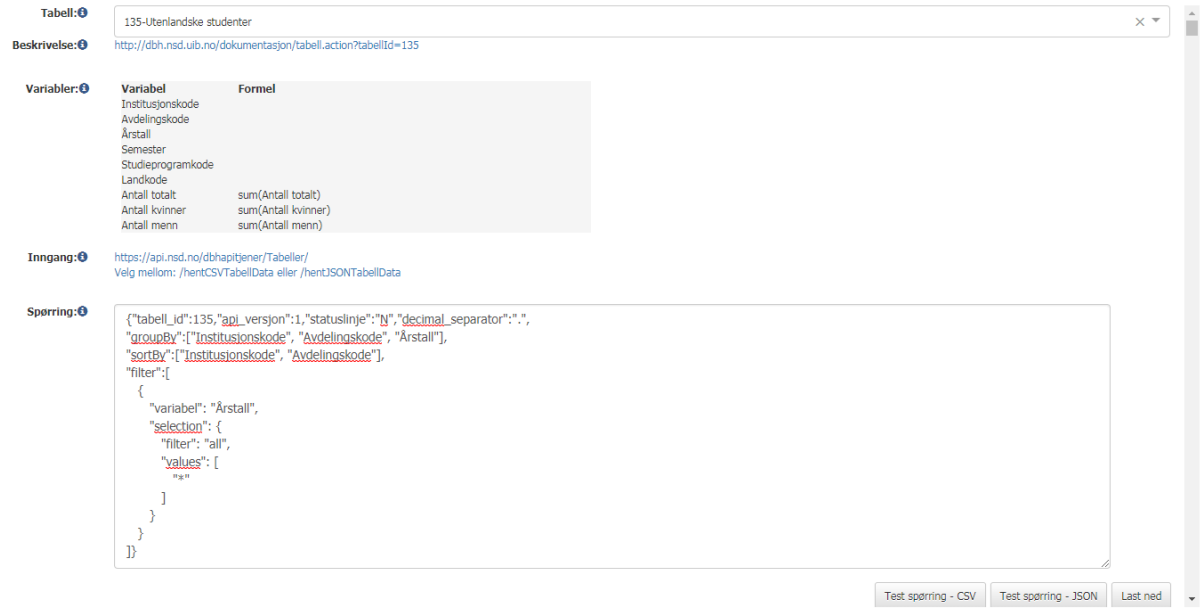

```
{"tabell_id":132,"api_versjon":1, "statuslinje":"N", "kodetekst":"N","decimal_separator":".",
"groupBy":["Institusjonskode", "Avdelingskode", "Årstall"],
"sortBy":["Institusjonskode", "Avdelingskode", "Årstall"],
"filter":[
  {
    "variabel": "Institusjonskode",
    "selection": {
      "filter": "all",
      "values": [
        "*"
      ]
    }
  },
  {
    "variabel": "Avdelingskode",
    "selection": {
      "filter": "all",
      "values": [
        "*"
      ]
    }
  },
  {
    "variabel": "Årstall",
    "selection": {
      "filter": "top",
      "values": [
        "2"
      ]
    }
  }
]}
```


Import fra 'Last ned CSV resultat'.
Velge 'Data' – 'Fra tekst' – 'Importer' – 'Data med skilletegn' – 'Filopprinnelse: Unicode (Utf-8)' 'Skilletegn – Semikolon'. Dette er standardmettoden for import av datafilen.

Ved å velge importer som tekst på feltnivå, vil feltet vise slik som i datafilen.

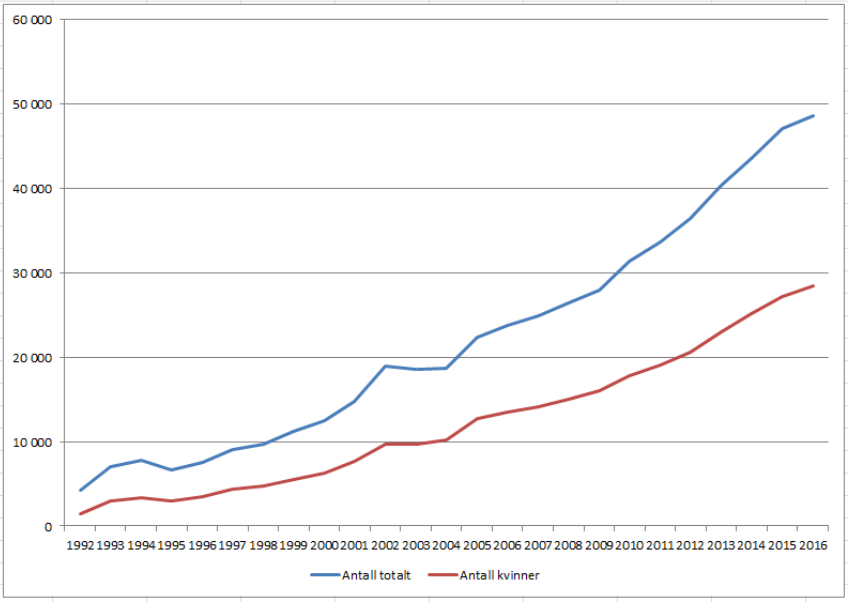
## Excel – Overføring av datasett via API klient.

Eksempel på spørring:




Disse dataene kan vi legge over i Excel.

| Årstall | Antall totalt | Antall kvinner |
|---|---|---|
| 1992 | 4246 | 1515 |
| 1993 | 7077 | 2987 |
| 1994 | 7761 | 3353 |
| 1995 | 6686 | 3034 |
| 1996 | 7582 | 3495 |
| 1997 | 9090 | 4319 |
| 1998 | 9644 | 4799 |
| 1999 | 11236 | 5500 |
| 2000 | 12431 | 6276 |
| 2001 | 14732 | 7649 |
| 2002 | 19013 | 9642 |
| 2003 | 18560 | 9759 |
| 2004 | 18745 | 10166 |
| 2005 | 22403 | 12715 |
| 2006 | 23819 | 13541 |
| 2007 | 24962 | 14148 |
| 2008 | 26480 | 15026 |
| 2009 | 27982 | 16082 |
| 2010 | 31436 | 17854 |
| 2011 | 33664 | 19089 |
| 2012 | 36489 | 20665 |
| 2013 | 40369 | 22995 |
| 2014 | 43576 | 25118 |
| 2015 | 47157 | 27181 |
| 2016 | 48648 | 28459 |




Her viser vi tabell for utenlandske kvinnelige studenter fra 1992 til 2016.

## Sammenligning DBH Api og DBH Rapporter :

I noen tilfeller ser vi at det er differanser mellom summene som vises i DBH rapportdel og de summene som vises i API.

De vanligste årsakene er:

- Valg/avkryssinger i DBH rapport.
  I API/Klient henter bruker ofte hele tabellen hun/han ønsker å se.
  I rapportdelen gjøres valg som fører til at data blir filtrert bort. Derfor kan det oppstå større differanser.
  I API oppretter vi et forslag til spørring som bruker kan endre og legge inn nye filter. Forslaget er basert på at alle data hentes. Bruker kan selv legge inn begrensninger på data som skal leveres.

- Valg av avrunding i DBH rapporter.
  Av og til ser vi differanser på 1(f.eks) mellom API og Rapport.
  Dette skyldes mest sannsynlig avrunding av data i Rapportdelen.
  I API avrunder vi ikke data.
  Om du er pålogget på Statstikksiden, kan du velge å ta bort avrunding.
  Da vil små differanser vanligvis forsvinne.

- Pålogging i API.
  Om du ikke er definert for å se persondata (små tall) for egen institusjon, må du opprette bruker. Velg Logg inn og opprett bruker i API klient bildet.
  Send deretter beskjed til DBH om at vi må opprette autorisering til å se små tall
  Logg deretter inn og spør på nytt.

Om nødvendig kan du teste med å sette "statuslinje":"J". Da vil du få beskjed om antall skjerminger.

## O2AUTH bruk i PHYTON.

```
auth = (“my-ID”, “my-secret”)  # Bytte ut disse
post_data = {"grant_type": "client_credentials"}
response = requests.post("https://sso.nsd.no/oauth/token", auth=auth, data=post_data)
response_object = response.json()
api_token = response_object["access_token"]
```

```
library(rjson)
library(jsonlite)
library(httr)
library(tidyverse)
```

## ‘R’ – programmering

Når vi henter data programatisk, krever vi pålogging. D.v.s maskinen må være definert hos NSD. Det må hentes et logon token og dette vil vare en time. Progammerer må derfor ordnedette for eksempel med en loop som gjør at programmet for henting våkner en gang i timen.
Det må også taes hensyn til kodesett når det gjelder spørringen (Windows/Dos/Unix).

### Eksempel1 med ‘R’ programmering

Dette eksempelet bruker ikke token pålogging. Tar ikke hensyn til norsk tegnsett.

```
#Eksempel med httr

library(foreign)
library(jsonlite)
library(httr)

library(rjstat)

options(encoding="UTF-8")

#Henter query
setwd(file.path("C:/Sync/Fak/2018/APIer/queryBody"))

resultat<- POST(url="https://api.nsd.no/dbhapitjener/Tabeller/hentJSONTabellData",
          add_headers("Content-Type: application/json"),
          body=upload_file('060-querybody2.json'),
          encode='json')

resultat_data <- fromJSON(content(resultat, "text"))
```

```
setwd(file.path("C:/Sync/Fak/2018/APIer/Result"))

#Fjerner linje 1 i data som inneholder Statuslinje
resultat_data = resultat_data[-1,]

#Fjerner alle variable med status i navnet
#resultat_data_clean<-resultat_data[ , !names(resultat_data) %in% c("status")]

#Skriver til en  csv-fil
write.csv(resultat_data_clean, file = "Tabell_060.csv")
```

## Eksempel2 - 'R' programmering

Dette eksempelet bruker ikke token pålogging. Tar hensyn til norsk tegnsett.

```
## DBH-API tabeller i jsnon format ----


dbh_api_tabell <- function(spørring,
                           tabellformat = "json",
                           status = TRUE,
                           api_versjon = 1,
                           decimal_mark = locale()$decimal_mark,
                           stringsAsFactors = FALSE) {
 tabellformat = str_to_upper(tabellformat)
 if (tabellformat!="JSON")
  warning(str_c(tabellformat, " er ikke et gyldig format, bruker JSON i stedet"))
 post_body = rjson::toJSON(
  c(list(
   api_versjon = api_versjon,
   statuslinje = "N",
   decimal_separator = decimal_mark),
   spørring))
 resultat <-
  POST(url = "https://api.nsd.no/dbhapitjener/Tabeller/hentJSONTabellData",
       add_headers(`Content-Type` = "application/json"),
       body = post_body,
       encode = "json")


 json_resultat <- fromJSON(content(resultat, "text"))

 resultat_data = json_resultat[-1,]
 resultat_data<-resultat_data[,!names(resultat_data)%in% c("status")]


}


## Tegn utenfor ASCII virker ikke i spørringene (via rjson::toJSON) når de
```

```
## tastes rett inn, men virker når de tastes inn via Unicode-koden (selv om
## resultatet i konsollen vises likt). Virker heller ikke når de kodes via
## funksjoner som utf8::utf8_encode(as.character(s), utf8 = FALSE)) eller
## stringi::stri_escape_unicode(s)

kod_norske_tegn <- function(s) {
  str_replace_all(s,
                  c(
                    "Æ" = "\u00c6",
                    "Ø" = "\u00d8",
                    "Å" = "\u00c5",
                    "æ" = "\u00e6",
                    "ø" = "\u00f8",
                    "å" = "\u00e5"
                  ))
}

## Funksjon for å lage spørring i DBH-API-format ----

dbh_api_spørring <- function(tabell_id,
                             filter_variabler,
                             filter_koder,
                             filter_verdier,
                             group_by = list(),
                             sort_by = list()) {
  var_len <- length(filter_variabler)
  filter_variabler<-map(rep_len(filter_variabler, var_len), kod_norske_tegn)
  filter_koder<-map(rep_len(filter_koder, var_len), kod_norske_tegn)
  filter_verdier<-map(rep_len(filter_verdier, var_len), kod_norske_tegn)
  group_by =  map(group_by, kod_norske_tegn)
  sort_by =  map(sort_by, kod_norske_tegn)

  list(tabell_id = tabell_id,
       groupBy = group_by,
       sortBy = sort_by,
       filter = pmap(
         list(filter_variabler,
              filter_koder,
              filter_verdier),
         function(filter_var, filter_kode, filter_verdi) {
           list(variabel = filter_var,
                selection = list(filter = filter_kode,
                                 values =
                                   if(length(filter_verdi) == 1)
                                     list(filter_verdi) else filter_verdi))
         }))
}
```

#Studiepoeng

```
studiepoeng<-dbh_api_tabell(dbh_api_spørring(900,"Årstall", "top", "3"))


#Kandidatar
kandidatar<-dbh_api_tabell(dbh_api_spørring(907,"Årstall", "item", "2016","2017"))


#Institusjontype

institusjon_type<-dbh_api_tabell(dbh_api_spørring(287,"Institusjonstypekode", "all","*"))

#Doktorgrad

doktorgrader<-dbh_api_tabell(dbh_api_spørring(101,group_by = list("Institusjonskode",  "Årstall"),
                                sort_by =list ("Institusjonskode"),filter_variabler = "Årstall", filter_kode =
"top", filter_verdier = "3" ))

# DOktorgrad samarbeid
Doktorgrad_samarbeid<-dbh_api_tabell(dbh_api_spørring(100,"Årstall", "top", "3"))


#Utveksling studenter

utveksling<-dbh_api_tabell(dbh_api_spørring(142,
group_by=list("Institusjonskode","Årstall","Type","Utvekslingsavtale"),
                                sort_by ="Institusjonskode",filter_variabler = "Årstall", filter_kode = "top",
filter_verdier ="3"))

View(utveksling)
```

## Eksempel3 med 'R' programmering

Dette eksempelet bruker token pålogging. Tar hensyn til norsk tegnsett.

```
## Laster pakker ----
library(rjson)
library(jsonlite)
library(httr)
library(magrittr)
library(stringr)
library(lubridate)
library(tidyverse)


## Håndterer token-innlogging ----

# Globale variabler for nåværende token og utløpstid
dbh_api_token_utløpstid <- Sys.time()
dbh_api_token_innhold <- ""
```

```
# Henter nytt token
dbh_api_token_hent_nytt <- function() {
 res <-
  POST(url = "https://sso.nsd.no/oauth/token",
       authenticate(user = "sso_id",
                    password = "sso_secret"),
       body = list(grant_type = "client_credentials"),
       encode = "form") %>%
  content("text") %>%
  fromJSON()
 return(res$access_token)
}

# Returnerer nåværende token fra global variabel, eller henter nytt token hvis
# utløpt
dbh_api_token <- function() {
 t <- Sys.time()
 if (t >= dbh_api_token_utløpstid) {
  walk2(
   str_c("dbh_api_token_",
         c("utløpstid", "innhold")),
   list(t + 3600,
        dbh_api_token_hent_nytt()),
   assign,
   env = .GlobalEnv
  )
 }
 return(dbh_api_token_innhold)
}


## Funksjon for å hente tabeller i DBH-API ----

dbh_api_tabell <- function(spørring,
                           tabellformat = "json",
                           status = TRUE,
                           api_versjon = 1,
                           decimal_mark = locale()$decimal_mark) {
 tabellformat = str_to_upper(tabellformat)
 if (! tabellformat %in% c("CSV", "JSON"))
  warning(str_c(tabellformat, " er ikke et gyldig format, bruker JSON i stedet"))
 post_body = rjson::toJSON(
  c(list(
   api_versjon = api_versjon,
   statuslinje = if_else(status, "J", "N"),
   decimal_separator = decimal_mark),
   spørring))
 res <-
  POST(url = sprintf("https://api.nsd.no/dbhapitjener/Tabeller/hent%sTabellData", tabellformat),
       add_headers(`Content-Type` = "application/json",
                   Authorization = str_c("Bearer ", dbh_api_token())),
```

```
      body = post_body,
      encode = "json") %>%
   content("text")
 if (tabellformat == "JSON") {
   json_res <- jsonlite::fromJSON(res)
   status = drop_na(as_tibble(json_res$status))
   # fjerner første kolonne (som er$status og inneholder nøstet df) og første rad (som er fylt i $status,
og tom i øvrige kolonner)
   data = as_tibble(json_res[-1,-1])
 } else {
   status = NULL
   data = read_delim(res,
                     delim = ";",
                     col_types = cols(.default = col_character()),
                     locale = locale(decimal_mark = decimal_mark),
                     na = "",
                     trim_ws = TRUE
   )
 }
 return(list(status = status,
             data = data))
}


## Tegn utenfor ASCII virker ikke i spørringene (via rjson::toJSON) når de
## tastes rett inn, men virker når de tastes inn via Unicode-koden (selv om
## resultatet i konsollen vises likt). Virker heller ikke når de kodes via
## funksjoner som utf8::utf8_encode(as.character(s), utf8 = FALSE)) eller
## stringi::stri_escape_unicode(s)

kod_norske_tegn <- function(s) {
 str_replace_all(s,
                 c(
                   "Æ" = "\u00c6",
                   "Ø" = "\u00d8",
                   "Å" = "\u00c5",
                   "æ" = "\u00e6",
                   "ø" = "\u00f8",
                   "å" = "\u00e5"
                 ))
}

## Funksjon for å lage spørring i DBH-API-format ----

dbh_api_spørring <- function(tabell_id,
                             filter_variabler,
                             filter_koder,
                             filter_verdier,
                             group_by = list(),
                             sort_by = list()) {
 var_len <- length(filter_variabler)
 filter_koder <- rep_len(filter_koder, var_len)
```

```
  filter_verdier <- rep_len(filter_verdier, var_len)
  filter_variabler <- map(filter_variabler, kod_norske_tegn)
  filter_koder <-  map(filter_koder, kod_norske_tegn)
  filter_verdier <-  map(filter_verdier, kod_norske_tegn)
  group_by =  map(group_by, kod_norske_tegn)
  sort_by =  map(sort_by, kod_norske_tegn)
  list(tabell_id = parse_integer(tabell_id),
      groupBy = group_by,
      sortBy = sort_by,
      filter = pmap(
        list(filter_variabler,
            filter_koder,
            filter_verdier),
        function(filter_var, filter_kode, filter_verdi) {
          list(variabel = filter_var,
              selection = list(filter = filter_kode,
                               values =
                                 if(length(filter_verdi) == 1)
                                   list(filter_verdi) else filter_verdi))
        }))
}

# Hjelpefunksjon som laster ned spørringer til filer (separat for data og status)

dbh_hent_spørring_til_fil <- function(spørringer, datanavn) {
  cat("\n", datanavn, "\n")

  pb <- progress_estimated(length(spørringer))

  data_l <-
    spørringer %>%
    map(~{
      pb$tick()$print()
      dbh_api_tabell(.)
    }) %>%
    transpose()

  status <-
    data_l$status %>%
    bind_rows()

  names(data_l$data) <- status$leveransenr

  data <-
    data_l$data %>%
    `[`(!is.na(status$leveransenr)) %>%
    bind_rows(.id = "leveransenr")

  write_tsv(data, str_c(datanavn, ".csv"), na = "")
  write_tsv(status, str_c(datanavn, "_status.csv"), na = "")

  pb$stop()$print()
```

```
}

## Håndterer institusjoner ----

institusjoner <-
  dbh_api_tabell(dbh_api_spørring(211, "Institusjonskode", "all", "*"))

institusjoner_status <- institusjoner$status
institusjoner <- institusjoner$data

# Leser manuell liste over fusjoner
institusjonsendringer <-
  read_tsv("institusjonsendringer.tsv", na ="")

# Hjelpefunksjon for å finne siste institusjon hvor det er rekker av fusjoner.
# Foreløpig en forenkling som ser bort fra gyldighetsinformasjon i DBH og dato
# for sammenslåing.
finn_nyeste <- function(aktuell_ny, nye, gamle) {
  if (! aktuell_ny %in% gamle) {
    return(aktuell_ny)
  } else {
    return(finn_nyeste(nye[match(aktuell_ny, gamle)], nye, gamle))
  }
}

institusjonsendringer <- institusjonsendringer %>%
  mutate(
    institusjonskode_nyeste =
      map_chr(
        institusjonskode_ny,
        finn_nyeste,
        institusjonskode_ny,
        institusjonskode_gammel
      )
  )

institusjoner <- institusjoner %>%
  mutate(institusjonskode_nyeste =
          institusjonsendringer$institusjonskode_nyeste[match(Institusjonskode,
                                                              institusjonsendringer$institusjonskode_gammel)],
         institusjonskode_nyeste = if_else(is.na(institusjonskode_nyeste),
                                           Institusjonskode,
                                           institusjonskode_nyeste))


# Behandler metadata og spørringer for utvalg av tabeller ----

tabeller <-
  tribble(~tab, ~tabell_id,
          "studiepoeng", 900,
          "doktorgrader", 101,
          "publisering", 373,
```

```
        "utveksling", 142)


# Legger til info til spørringer (group_by)

tabeller <-
  full_join(tabeller,
          tribble(
            ~ tabell_id,
            ~ group_by_spørring,
            373, list(
              "Institusjonskode",
              "Årstall",
              "Kode for publikasjonsform",
              "Kode for publiseringskanal",
              "Kode for type publiseringskanal"
            ),
            142, list(
              "Institusjonskode",
              "Årstall",
              "Type",
              "Utvekslingsavtale"
            ),

            101, list(
              "Institusjonskode",
              "Avdelingskode",
              "Årstall",
              "Semester",
              "Kvalifikasjonskode",
              "Studieprogramkode"  ,
              "Finansieringskildekode"
            )
          ),
          by = "tabell_id")

# Bytter ut NULL med list() (p.t. bare aktuelt for group_by)
tabeller <- tabeller %>%
  mutate_all(funs(ifelse(map_lgl(., is.null),
                         map(seq_along(.),
                         ~list()),
                         .)))

spørringer <-
  tabeller %>%
  pmap(function(tabell_id, group_by_spørring, tab, ...) {
    map(institusjoner$Institusjonskode,
        ~dbh_api_spørring(tabell_id,
                          filter_variabler = "Institusjonskode",
                          filter_kode = "item",
                          filter_verdier = .,
                          group_by = group_by_spørring)) %>%
```

```
    list(.) %>%
    setNames(., tab)
 }) %>%
 unlist(recursive = FALSE)


# Laster ned alle tabellene i spørringer/tabeller og skriver til fil

walk2(spørringer,
    names(spørringer),
    dbh_hent_spørring_til_fil)

# Leser tabeller fra fil til minne

walk(names(spørringer),
   ~{
     str_c(., c("", "_status")) %>%
       walk(~assign(.,
                read_tsv(str_c(., ".csv"),
                      na = "",
                      col_types = cols(.default = col_character())),
                envir = .GlobalEnv))
   })


# Sammenligne API-data med leveranse til finansieringssystemet ----

studiepoeng_agg <-
 studiepoeng %>%
 filter(str_to_upper(Studentkategori) == "S") %>%
 rename(kategori = `Finmodekode emne`,
       indikatorverdi = `Ny produksjon egentfin`) %>%
 mutate(kategori = str_to_upper(kategori),
       indikatorverdi = parse_number(indikatorverdi)) %>%
 group_by(Årstall, Institusjonskode, kategori) %>%
 summarise_at("indikatorverdi", sum) %>%
 ungroup() %>%
 mutate(indikator = "studiepoeng")

utveksling_agg <-
 utveksling %>%
 mutate_at(c("Type", "Utvekslingsavtale"), str_to_upper) %>%
 filter(Utvekslingsavtale != "INDIVID") %>%
 select(Årstall,
       Institusjonskode,
       Type,
       kategori = Utvekslingsavtale,
       indikatorverdi = `Antall totalt`) %>%
 mutate(
  indikator = "utveksling",
  indikatorverdi = parse_number(indikatorverdi),
  kategori =
```

```
    case_when(
      kategori == "ERASMUS+" &
       Type == "NORSK" ~ "Erasmus+",
      TRUE ~ "")
  )

doktorgrader_agg <-
  doktorgrader %>%
  mutate(indikator = "doktorgrader",
         kategori = "",
         indikatorverdi = parse_number(`Antall totalt`)) %>%
  group_by(Årstall, Institusjonskode, indikator, kategori) %>%
  summarise_at("indikatorverdi", sum) %>%
  ungroup()

publisering_agg <-
  publisering %>%
  mutate(indikator = "publisering",
         kategori = "",
         indikatorverdi = parse_number(Publiseringspoeng)) %>%
  group_by(Årstall, Institusjonskode, indikator, kategori) %>%
  summarise_at("indikatorverdi", sum) %>%
  ungroup()

samlet_agg_17 <-
  bind_rows(studiepoeng_agg,
            utveksling_agg,
            doktorgrader_agg,
            publisering_agg) %>%
  filter(Årstall == "2017") %>%
  left_join(select(institusjoner, Institusjonskode, institusjonskode_nyeste),
            by = "Institusjonskode") %>%
  group_by(institusjonskode_nyeste,
           indikator,
           kategori) %>%
  summarise_at("indikatorverdi", sum) %>%
  ungroup()

# Les finsysdata-leveranse i Excel til finsys_data

source("finsys_data_excel.R")

fasit_utvalg <-
  finsys_data %>%
  rename(institusjonskode_nyeste = instkode) %>%
  filter(indikator %in% c("studiepoeng", "utveksling", "doktorgrader", "publisering")) %>%
  select(institusjonskode_nyeste, indikator, kategori, indikatorverdi)

sjekk_fasit <-
  full_join(samlet_agg_17,
            fasit_utvalg,
            by = c("institusjonskode_nyeste",
```

```
            "indikator",
            "kategori")) %>%
  rename(indikatorverdi_api = indikatorverdi.x,
         indikatorverdi_fasit = indikatorverdi.y) %>%
  mutate(diff = indikatorverdi_api - indikatorverdi_fasit)

avvik_api_fasit <-
  sjekk_fasit %>%
  filter(kategori %in% c(LETTERS[1:6], "", "Erasmus+"),
         !is.na(diff),
         round(diff, 5) != 0) %>%
  left_join(select(institusjoner, institusjonskode_nyeste = Institusjonskode, Institusjonsnavn),
            by = "institusjonskode_nyeste") %>%
  arrange(indikator, desc(abs(diff)))

export(avvik_api_fasit, "avvik_dbh_api.xlsx")
```