Hippo, Statistik och TPDB

# Bakgrund

# Komponenter

## Översikt

Följande källkodsfiler ingår:

./images/ajax-loader.gif

./lib/statdbupdate.ini

./lib/statdb.ini

./statdb/leolib\_debug.php

./statdb/download.js

./statdb/leolib\_sql.php

./statdb/ViewIntegrationOne.sql

./statdb/statdb.js

./statdb/statdb.php

./statdb/statdbupdate.php

./statdb/statdb\_2018-01-10.sql

./statdb/ViewIntegrationsDescriptions.sql

./statdb/ViewIntegrationsMulti.sql

./statdb/download.js.externs

./statdb/ViewIntegrationTwo.sql

./views/statistics/statistics1.js

./views/statistics/google\_visualization\_api.js.externs

./views/statistics/statistics.html

./views/searchview/searchview5.js

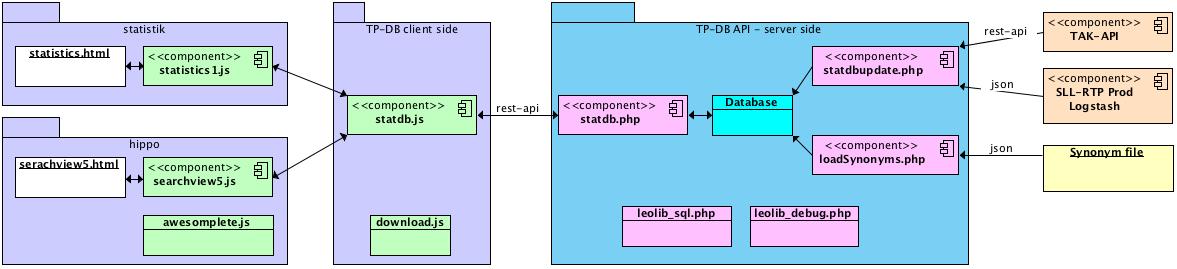
./views/searchview/awesomplete.js.externs

./views/searchview/awesomplete.js

./views/searchview/awesomplete.css

./views/searchview/searchview5.html

Följande diagram ger en översikt över de ingående komponenterna.



## Frontend sökvy – hippo

Den aktuella versionen av "hippo", dvs webbapplikation för att navigera och söka sig fram bland integrationer.

Den består av följande källkodsfiler:

./views/searchview/searchview5.js

./views/searchview/awesomplete.js.externs

./views/searchview/awesomplete.js

./views/searchview/awesomplete.css

./views/searchview/searchview5.html

searchview5.html är en minimal html-fil. Dess huvudsakliga funktion är att agera startpunkt för applikationen. Den laddar de javascript som behövs samt definierar ett antal platser där scripten kan lägga ut sin information.

Den logik som bygger upp hippo-gränssnittet är definierad i searchview5.js. Den använder sig av statdb/statdb.js för att hämta information från databasen.

.externs-filen innehåller deklarationer som krävs för clousure compiler.

awescomplete är ett extern javascriptbiblotek som används i sökrutorna. Mer information finns på hemsidan <http://leaverou.github.io/awesomplete/>. Dess CSS har modifierats så att bakgrundsfärgen skall överensstämma med huvudsidan.

## Frontend statistikvy

Statistikvyn är uppbygg av följande filer:

./views/statistics/statistics1.js

./views/statistics/google\_visualization\_api.js.externs

./views/statistics/statistics.html

statistics.html är en minimal html-fil. Dess huvudsakliga funktion är att agera startpunkt för applikationen. Den laddar de JavaScript som behövs samt definierar ett antal platser där scripten kan lägga ut sin information.

Huvudlogiken finns i statistics1.js. Den använder sig av statdb/statdb.js för att accessa databasen och hämta statistikinformationen. För att rendera diagrammen nyttjas [Google chart library](https://developers.google.com/chart/).

## TP-DB client side (frontend gemensam)

Denna komponents används av alla vyer som baseras på hippo-databasen (idag sökfunktionen och statistiken).

./statdb/statdb.js

./statdb/download.js

./statdb/download.js.externs

**statdb.js** innehåller funktioner för att interagera med backendkod och databas. Här återfinns även datastrukturer

som håller det data som hämtats ner. Ett centralt objekt är Filter() där man definierar de villkor som skall gälla när data hämtas.

**download.js** innehåller funktioner för spara fil med data som kommer från datastrukturer i browsern (dvs inte en traditionell nedladdning från en server). ".externs" krävs för att kunna göra en komprimering med Closure compiler (besrkivs i byggrutinen).

## TB-DB server side

Den information som hanteras och visas i vyerna lagras i en relationsdatabas. Ursprungligen utvecklades logiken i en MySQL. I produktion används dock MariaDB, och även Amazon Aurora har använts under utveckling och test. De två senare är kompatibla med MySQL.

Förutom databasen finns ett antal PHP-script för att accessa och uppdatera databasinformationen.

## Databasschema

Databasen består idag av 13 tabeller och tre vyer. 11 av tabellerna används för att spara information från de anslutna TAK-arna.

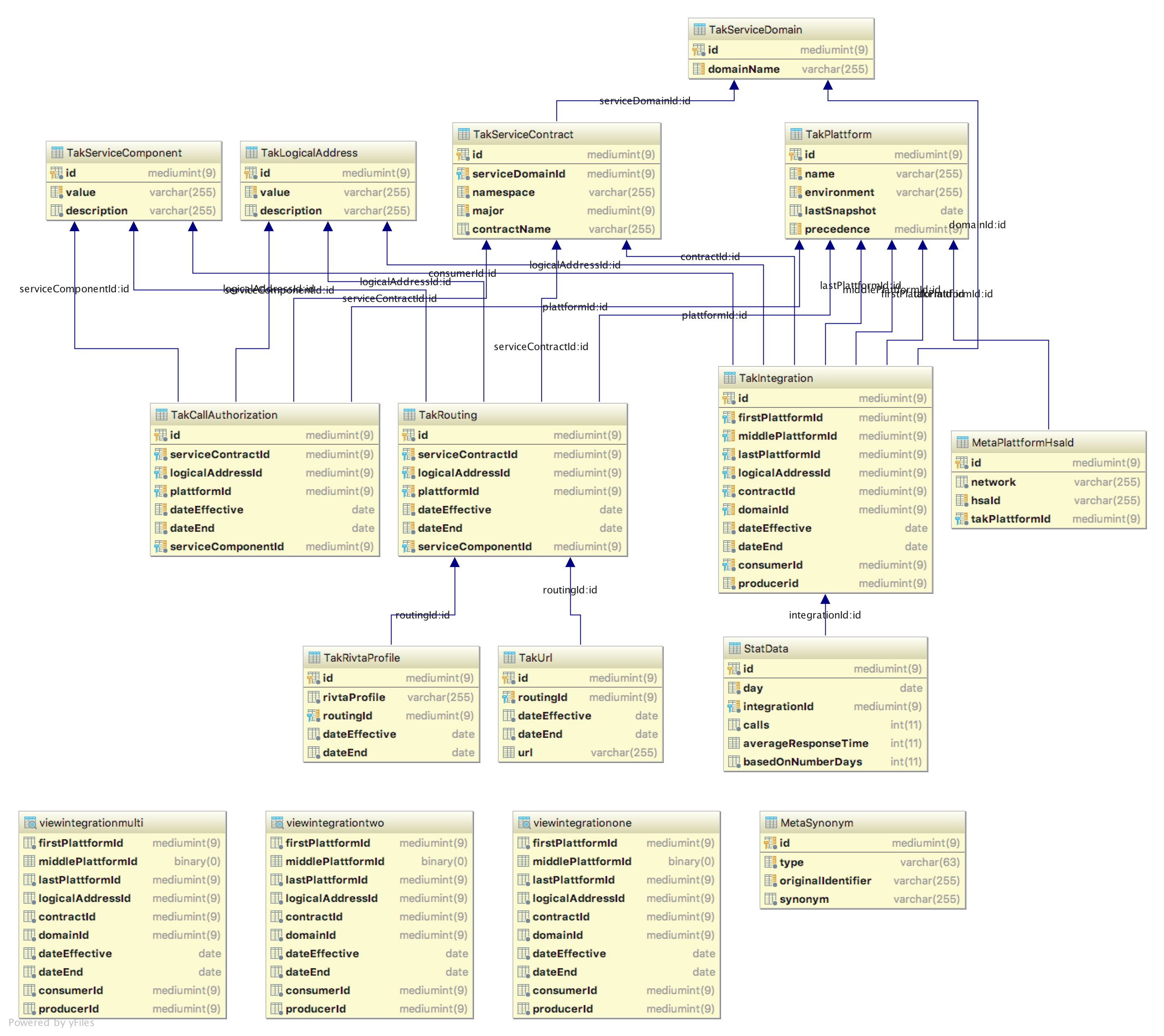
Schemat stödjer historiskt data vilket gör det möjligt att läsa ut information som visar hur en TAK var beskaffad ett specifikt datum.

En tabell lagrar synonymer, och en tabell är dedikerad till att lagra grunddata för statistik.

Den senare är förberedd för att spara antal anrop per dygn samt genomsnittlig svarstid per dygn.

Vyerna används enbart i samband med att databasen uppdateras från TAK-api. De samverkar till att extrahera information om integrationer (som kan vara över flera tjänsteplattformar). Resultatet lagras i en tabell som sedan används av vyerna. På detta sätt erhålls kortare svarstider än att nyttja vyerna direkt.

Databasschemat ser ut som:



Varje tabell och vy) har ett prefix som visar dess roll i informationshanteringen.

* **Tak** - lagrar information från TAK-arna
* **Meta** - innehåller konfigurationsinformation som endast undantagsvis förändras, och i så fall sker det manuellt
* **Stat** - innehåller statistikunderlag
* **View** - en vy

**TakPlattform**

Innehåller en rad per tjänsteplattform/TAK som hanteras. Förutom namn sparas även tidpunkt för senaste uppdatering samt prioritetsordning vad gäller hur mycket man skall lita på de beskrivande texterna som finns.

**MetaPlattformHsaId**

Information som kopplar ett HSA-id till en tjänsteplattform. Det gäller de HSA-id som tjänsteplattformen visar när den uppträder som tjänstekonsument och/eller tjänsteproducent. Även information om vilket nätverk som detta HSA-id används på lagras.

**TakServiceComponent**

Lagrar HSA-id och beskrivande text för tjänstekomponenter (tjänstekonsumenter och tjänsteproducenter). Beskrivningen är alltid den senaste som återfinns för detta objekt i den tak med högst prioritet (enligt TakPlattform).

**TakLogicalAddress**

Lagrar logisk adress och dess beskrivning. Beskrivningen är alltid den senaste som återfinns för detta objekt i den tak med högst prioritet (enligt TakPlattform).

**TakServiceDomain**

Det tekniska domännamnet. Det är en subset av namnrymnden för tjänstekontrakten.

**TakServiceContract**

Tjänstekontraktets namnrymnd, namn och majorversion. Dessutom en referens till aktuell tjänstedomän i TakServiceContract.

**TakCallAuthorization**

Här lagras information om anropsbehörigheter. Det är en "kopplingstabell" som definierar en behörighet genom att via främmande nycklar referera en tjänstekonsument (tjänstekomponent), en logisk adress och ett tjänstekontrakt. En anropsbehörighet hör dessutom till en specifik tjänsteplattform, också definierat via en främmande nyckel.

Denna tabell definierar även historik. En anropsbehörighet existerar under en specifik tidsperiod. Fältet *dateEffective* innehåller det första datum som behörigheten existerar, och *dateEnd* den sista.

**TakRouting**

Här lagras information om vägval. Det är en "kopplingstabell" som definierar ett vägval genom att via främmande nycklar referera en tjänsteproducent (tjänstekomponent), en logisk adress och ett tjänstekontrakt. Ett vägval hör dessutom till en specifik tjänsteplattform, också definierat via en främmande nyckel. Till varje vägval är också kopplat en URL och en RIVTA-profil.

Denna tabell definierar även historik. En anropsbehörighet existerar under en specifik tidsperiod. Fältet *dateEffective* innehåller det första datum som behörigheten existerar, och *dateEnd* den sista.

**TakUrl**

Innehåller den URL som gäller för ett specifikt vägval.

Denna tabell innehåller historik för att möjliggöra att identifiera förändringar av URL för ett specifikt vägval.

**TakRivtaProfile**

Kopplar en RIVTA-profil till ett vägval.

Denna tabell innehåller historik.

**TakIntegration**

I denna tabell definieras de faktiskt integrationerna, dvs koppling mellan tjänstekonsument och tjänsteproducent via en eller flera tjänsteplattformar. Denna information tillhandahålls inte av TAK-api. Istället genereras det upp i samband med den dygnsvisa uppdateringen av tabellerna ovan. Tabellen byggs upp av främmande nycklar mot upp till tre tjänsteplattformar (första, mitten, sista i en kedja), logisk adress, tjänstekontrakt, tjänstedomän, tjänstekonsument och tjänsteproducent. Dessutom har varje integration ett start- och ett slutdatum.

Denna information skulle i princip kunna selekteras dynamiskt. Det finns två huvudsakliga skäl till denna tabell:

1. Prestanda blir bättre när integrationerna för-genereras på detta sätt.

2. Integrationerna får ett unikt id i databasen. Detta id är konstant över tid. Det används som främmande nyckel i tabellen *StatData*.

**StatData**

Denna tabell innehåller en rad per \*TakIntegration\* och kalenderdag. I tabellen lagras antal anrop för denna integration aktuell dag (vilket kan vara ett genomsnitt av antal anrop per dag). Dessutom finns ett fält för genomsnittlig svarstid (som ännu inte används).

Det existerar bara rader för de integrationer där vi har tillgång till statistikinformation. I dagsläget (2018-01-19) gäller det för SLL-RTP-Prod från och med 2017-07-31.

**MetaSynonym**

Här lagras synonymer. I dagsläget finns sådana för tjänstekontrakt och tjänstekomponenter (tjänstekonsumenter och -produceter).

**Vyerna**

Vyerna söker fram (definierar) integrationer. En integration är i grunden en koppling av en anropsbehörighet och ett vägval som har samma tjänstekontrakt och logisk adress i samma TAK och existerar vid samma tidpunkt. Om en integration pekar ut en annan tjänsteplattform som tjänsteproducent, och dess TAK refererar till den första tjänsteplattformen som tjänstekonsument, så hittar vi en integration över två plattformar.

## Databasuppdatering

All uppdatering av databasen sker via PHP-scripten *statdbupdate.php* och *loadSynonyms.php*.

**statdbupdate.php**

Detta script körs dygnsvis via ett cronjobb. Det gör två anrop mot TAK-api och hämtar samtliga *cooperations* och *serviceProductions*.

En *cooperation* motsvarar en anropsbehörighet i en av TAK-arna. Den innehåller all information som behövs även för de ingående delkomponenterna; logisk adress, tjänstekonsument, tjänstekontrakt och vilken TAK det gäller.

[  
 {  
 "id": 0,  
 "serviceConsumer": {  
 "id": 0,  
 "hsaId": "string",  
 "description": "string"  
 },  
 "logicalAdress": {  
 "id": 0,  
 "logicalAddress": "string",  
 "description": "string"  
 },  
 "serviceContract": {  
 "id": 0,  
 "name": "string",  
 "namespace": "string",  
 "major": "string",  
 "minor": "string"  
 },  
 "connectionPoint": {  
 "id": 0,  
 "platform": "string",  
 "environment": "string",  
 "snapshotTime": "string"  
 }  
 }  
]

Utifrån detta data uppdateras (vid behov) respektive tabell i databasen.

En *serviceProduction* motsvarar ett vägval. Det innehåller också komplett information för de ingående datamängderna:

[  
 {  
 "id": 0,  
 "physicalAddress": "string",  
 "rivtaProfile": "string",  
 "serviceProducer": {  
 "id": 0,  
 "hsaId": "string",  
 "description": "string"  
 },  
 "logicalAdress": {  
 "id": 0,  
 "logicalAddress": "string",  
 "description": "string"  
 },  
 "serviceContract": {  
 "id": 0,  
 "name": "string"  
 "namespace": "string",  
 "major": "string",  
 "minor": "string"  
 },  
 "connectionPoint": {  
 "id": 0,  
 "platform": "string",  
 "environment": "string",  
 "snapshotTime": "string"  
 }  
 }  
]

Utifrån denna information uppdateras tabeller för de ingående datamängderna (logisk adress, tjänstekomponent, tjänstekontrakt) samt vägvalstabellen, url-tabell och tabellen med RIVTA-profiler.

Det faktum att *cooperations* och *serviceProductions* innehåller komplett information gör att övriga API-anrop mot TAK-api inte är nödvändiga.

När grunddatatabellerna är uppdaterade enligt ovan sker en uppdatering av *TakIntegration*-tabellen. Indata till operationen är vyn *ViewIntegrationsMulti.sql*. Datat införs i TakIntegration, och datumfälten uppdateras. Ett slutdatum identifieras genom att en integration i TakIntegration inte längre genereras från vyn. Här är det också viktigt att inte förändra id-fältet för befintliga integrationer, eftersom tabellen *StatData* använder det som främmande nyckel.

Det sista steget i uppdateringen är att statistikdata läses in och lagras i *StatData*. Det mappas mot integrationerna i *TakIntegration*.

**loadSynonyms.php**

Detta script läser en fil med synonymdefinitioner och populerar tabellen MetaSynonym. Indata är en CSV-fil med följande kolumner:

* **type** (tillåtna typer är "serviceContract" och "serviceComponent"
* **originalIdentifier** (för tjänstekontrakt är det namespace, för tjänstekomponenter dess HSA-id)
* **synonym** (den synonym som skall kunna visas istället för beskrivningsfältet)

Eventuellt övriga kolumner läses inte. Om en synonym i filen är tom så tas den även bort från tabellen.

Filen skall ha namnet "MetaSynonym.csv".

Detta kommando är inte schemalagt utan måste exekveras manuellt i samband med att filen uppdaterats.

## Backend internt REST-API

**statdb.php** exekverar på servern i anslutning till databasen.

Den består till största delen av SQL SELECT satser som hämtar ut data ur databasen.

Datat formateras och returneras i en JSON-struktur.

För vissa anrop komprimeras JSON-strukturen för att minska datamängden och öka den upplevda prestandan.

API’t tillhandahåller följande anrop.

### dates

**Exempelanrop:** https://<server>/statdb.php/api/v1/dates

**Svarsschema:**

{

"dates": {

"integrations": [

"2018-04-25",

…

],

"statistics": [

"2018-04-22",

…

]

}

**Parameterar:** -

**Beskrivning:**

### plattforms

**Exempelanrop:** https://<server>/statdb.php/api/v1/plattforms

**Svarsschema:**

[

{

"id": 2,

"platform": "NTJP",

"environment": "QA",

"snapshotTime": "2018-04-25"

},

…

]

**Parameterar:** -

**Beskrivning:**

Anropet returnerar information om vilka tjänsteplattformars TAK-ar som finns i TP-DB.

## Backend Javascript-API

**statdb.js** är klientdelen av API.

Den definierar datastrukturer för att hantera integrations- och statistikdata och gör det tillgängligt för de JAVASCRIPT-applikationer som vill utnyttja datat.

Detta script tillhandahåller på så sätt en högre abstraktionsnivå, samt diverse stödfunktioner som är användbara för applikationsutvecklaren.

# Utveckling

## Installera utvecklingsmiljö

```

git clone URL directory

git co -b develop

```

## Databas

*mariadb* används som databas i produktion, och det fungerar också utmärkt under utveckling. Den är ”i princip” helt kompatibel med mysql, även med kommandon och drivrutiner.

För utveckling är det enklaste är att utgå från en kopia av produktionsdatabasen. Det tas en backup varje natt mha mysqldump. Hämta över filen till utvecklingsmiljön.

För MACOS, installera mariadb enligt instruktionerna på sidan <https://mariadb.com/kb/en/library/installing-mariadb-server-pkg-packages-on-macos/>.

Nästa steg är att skapa användare, databas samt göra restore på den kopia av produktionsdatabasen som hämtats.

sudo mysql

GRANT ALL PRIVILEGES ON \*.\* TO ‘user’@'localhost' IDENTIFIED BY 'xxx';

create database skoview2 DEFAULT CHARACTER SET utf8;

mysql -u user -p -o skoview2 < mysql\_backup\_2018-04-11.sql

En förutsättning för ovanstående är (verkar vara) att *mydbuser* också är en användare på det system som kör databasen. I detta exempel alltså en unix-användare i MacOS.

*skoview2* är namnet på den databas som man gör restore på från filen *mysql\_backup\_2018-04-11.sql*.

Jag blev också tvungen att mecka med mariadbs sockets för att få ingång det från php. Se <https://stackoverflow.com/questions/4219970/warning-mysql-connect-2002-no-such-file-or-directory-trying-to-connect-vi>.

### Uppdateringsscript

Alla förändringar som skall ske av databasen inför en ny version skall ske i form av ett SQL-script. Det kan ex innehålla schemaförändringar och uppdateringar av Meta-tabellerna.

Ett exempel på ett sådant script:

*-- run with: mysql -u skoview -p -f skoview2 < Upgrade-5.1-to-5.2.sql  
-- Make sure the views exist and are updated***DROP VIEW IF EXISTS** viewintegrationmulti;  
**DROP VIEW IF EXISTS** viewintegrationtwo;  
**DROP VIEW IF EXISTS** viewintegrationone;  
**SOURCE** ViewIntegrationOne.sql;  
**SOURCE** ViewIntegrationTwo.sql;  
**SOURCE** ViewIntegrationMulti.sql;  
  
*-- Change the precedence span and insert new TAKs in TakPlattform***UPDATE** TakPlattform **SET precedence** = **precedence** \* 10 **WHERE precedence** < 10;  
**INSERT INTO** TakPlattform (**name**, **environment**, **lastSnapshot**, **precedence**) **VALUES** (**'LTD'**, **'PROD'**, **'2000-01-01'**, 25) **ON DUPLICATE KEY UPDATE id**=*LAST\_INSERT\_ID*(**id**);  
**INSERT INTO** MetaPlattformHsaId (**network**, **hsaId**, **takPlattformId**) **SELECT 'Sjunet'**, **'SE2321000180-1000'**, tp.**id FROM** TakPlattform tp **WHERE name** = **'LTD' ON DUPLICATE KEY UPDATE network**=**'Sjunet'**;

dd

## Test run

Local test run directly on the development branch

```

cd DOCUMENT\_ROOT

php -S localhost:4444 -t .

```

Then start a browser and point it to

```

http://localhost:4444/views/searchview/searchview5.html

```

## Versionshantering

## Byggprocess

The python based build program resides in ./buildstuff after cloning

### Build command

```

usage: build.py [-h] [-q] -t TARGET [-p PORT] -d DATABASE [-z] projectrootdir

Hippo build program...

positional arguments:

projectrootdir Project base folder

optional arguments:

-h, --help show this help message and exit

-q, --quiet Set to supress output (only rc will be set)

-t TARGET, --target TARGET

Mandatory. One of LOCAL, AWS or NOGUI

-p PORT, --port PORT If a port is specified a server instance listening to

the port and using the build is startet

-d DATABASE, --database DATABASE

Mandatory. One of DB-LOCAL, DB-AWS or DB-NOGUI

-z, --zip Remove source JS files and create a zip)

```

Local build, local db

```

/build.py -t LOCAL -d DB-LOCAL ../

```

Local build and start a httpd instance listening to port 3333

```

./build.py -p 3333 -t LOCAL -d DB-LOCAL ../

```

Then start a browser and point it to

```

http://localhost:3333/views/serachview/searchview5.html

```

## Test

# Driftsättning

### Elastic beanstalk

```

### Development server

cd /home/leo/Documents/data/Eternal/development/x1\_gitrepos/hippo/buildstuff

./build.py -z -t AWS -d DB-AWS ../

cd /home/leo/tmp/builddir/AWS\_DB-AWS

eb deploy

```

### NoGui server

För lösenord, se det vanliga stället nycklat över hippo NoGui.

Servernamn: skoview.i.pnet2.se / 192.36.9.137

Dokumentroot: /var/www/html/skoview/ Här lägger du dina web-prylar, php, js, bilder mm

Databasnamn: skoview

Databasuser: skoview

Det som är installerat:

RHEL7.3 inklusive

Apache 2.4

PHP 7.0

MariaDB

Vi har fixat alla SSL-konfigurationer och du kan som användare kika på dessa över /etc/httpd/conf.d

skoview.i.pnet2.se är nedlåst som alla våra maskiner, tråkigt men säkert :)

skoview.i.pnet2.se kommer inte åt Internet men Internet kommer åt skoview.i.pnet2.se

Du har full skrivaccess till Dokumentroot + din hemkatalog.

Deployment server

Connect to vpn skoview NoGui

```

kinit leoroj@PNET2.SE

klist

scp NOGUI\_DB-NOGUI.zip leoroj@skoview.i.pnet2.se:

ssh leoroj@skoview.i.pnet2.se

``

Installera den körbara koden

```

cd /var/www/html/skoview/

unzip ~/NOGUI\_DB-NOGUI.zip

mv lib/\*.ini ~/ini/

```

Hantera databasen

Det kan finnas ett behov av att hämta en uppdaterad databas, ex för att använda i utveckling eller installera på AWS. Använd i så fall följande kommando:

```

mysqldump --databases skoview2 --user=skoview --password > skoview2.sql

```

Detta finns i scriptet ~bin/dumpdb.sh.

Dumpfilen hämtas sedan med kommandot:

```

Installationsanvisning per release

Februari 2018

1. Bygg koden på vanligt sätt

1. Flytta över koden till skoview-server

1. Packa upp zipen i hemmappen

1. Addera information om nya TPs och HSA-idn genom att köra scriptet MetaPlattformHsaId\_update.sql

1. Skapa den nya tabellen MetaSynonym mha scriptet MetaSynonym\_create.sql

1. Placera synonymfilen MetaSynonym.csv i mappen där statistikatat ligger

1. Kör mysql-kommandot och ta bort de tre vyerna från databasen

1. Skapa om vyerna mha av de tre sql-scripten

1. Installera all kod (driftsätt)

1. Kör statdbupdate.php

1. Kör loadsynonyms.php

# Appendix A - Indata

## TAK-api

Nödvändigt grunddata för både hippo och statistikfunktionen hämtas från ett restbaserat TAK-api som tillhandahålls av Inera. API’t finns dokumenterat och tillgängligt via <http://api.ntjp.se/coop/doc/index.html>.

Det innebär att en organisation som vill presentera sin information via hippo och/eller statistikfunktionen först måste sätta upp en process för att föda TAK-api med grunddata. Ineras dokumentation för exportformat beskrivs här: <https://skl-tp.atlassian.net/wiki/spaces/SKLTP/pages/275251201/SKLTP+TAK+-+Export+av+takdata>. Kontakta Inera att komma överens och sätta upp en process för detta.

dd

## Statistikinformation

### Filformat

Filen med statistikinformation skall ha följande format:

1. {
2. "plattform": "SLL-QA",
3. "endDate": "2018-02-01",
4. "startDate": "2018-02-01",
5. "statistics": [{
6. "consumerId": "SE2321000016-9KP4",
7. "url": "https://qa.esb.ntjp.se:443/vp/infrastructure/eservicesupply/patientportal/AddMessageToPatientPortalInbox/1/rivtabp21",
8. "logicalAddress": "SE2321000016-A29F",
9. "namespace": "urn:riv:infrastructure:eservicesupply:patientportal:AddMessageToPatientPortalInboxResponder:1",
10. "calls": 19,
11. "averageResponseTime": 1234,
12. "originalConsumerId": "null"
13. }, {...
14. }]
15. }

**Filnamn**: *Organisation*-*Miljö*\_*Startdatum*--*Slutdatum*.STAT.json. Exemplet ovan skulle ha namnet: SLL-QA\_2018-02-01--2018-02-01.STAT.json

Innehållet är i JSON-format med följande attribut:

* **plattform:** Organisationer som äger plattformen, ett bindestreck, typ av miljö (PROD/QA/TEST). Exempel ”SLL-QA”, ”NTJP-PROD”.
* **startDate:** Första dagen för det tidsintervall denna statistikfil beskriver
* **endDate:** Sista dagen för det tidsintervall denna statistikfil beskriver
* **statistics:** En array bestående av följande attribut:
  + **consumerId:** HSA-id för den tjänstekonsument som direkt anropat denna plattform.
  + **originalConsumerId:** HSA-id för den ursprungliga tjänstekonsumenten enligt header *x-rivta-original-serviceconsumer-hsaid*. Detta fält kan enbart vara satt när consumerId refererar till en tjänsteplattform.
  + **logicalAddress:** Logisk adress
  + **namespace:** Hela tjänstekontraktets namnrymd
  + **url:** Den URL som aktuell tjänsteplattform använde för att anropa tjänsteproducenten.
  + **calls:** Totalt antal anrop för denna kombination av värden ovan under aktuellt tidsintervall.
  + **averageResponseTime:** Genomsnittlig svarstid för alla anrop för denna kombination av värden under aktuellt tidsintervall.

### Regelverk

1. Informationen om anrop och genomsnittlig svarstid nycklas i praktiken på en kombination av:

* day
* consumerId
* originalComsumerId
* logicalAddress
* namespace

1. En fil kan spänna över en eller flera dagar, vilket anges av attributen *startDate* och *endDate*. Den interna lagringen sker per dag. I det fall filen innehåller anrop för mer än en dag så kommer antal anrop att divideras med antal dagar.
2. Information för samma nyckel (se ovan) kan anges flera gånger (i flera filer). En fil med ett kortare tidsspanne (dvs endDate-startDate) antas ge en bättre noggrannhet och kommer att skriva över ett värde som kommer från en fil med längre tidsspann.
3. Filen kommer att läsas in en gång per dygn.

### Leverans av statistikfil

Leverans av statistikfil sker mha *sftp*. Kontakta förvaltningsobjekt Informationsinfrastruktur på SLL för att komma överens om detaljer för denna överföring. Adressen är *informationsinfrastruktur@sll.se.*

# Appendix B - Git

## Versionshantering för tpinfo

tpinfo är uppdelat i två git repos, tpinfo-frontend och tpinfo-backend. Båda finns tillgängliga på bitbucket:

* <https://bitbucket.org/hippo_team/tpinfo-frontend>
* <https://bitbucket.org/hippo_team/tpinfo-backend>

## Angreppssätt

Repositoriet på bitbucket betraktas som ”origin”.

Det finns två branches med evigt liv på origin:

* origin/master – motsvarar den version som fn är driftsatt i produktion
* origin/develop – motsvarar aktuell ”QA-version”, dvs stabil version som fn exkeverar i QA-miljön.

Varje gång en ny version är färdig för produktion så sammanfogas develop till master och taggas med versionsbeteckning.

Lokalt (på en utvecklarmaskin) kan det finnas ytterligare branches. Typiskt är:

* Branch för utveckling av ny funktionalitet
* Branch för att göra anpassningar inför driftsättning av ny release
* Branch för fixar

### Ny funktionalitet

I normalfallet finns denna branch enbart på utvecklarmaskiner.

Checkas ut från develop:

$ git checkout -b myfeature develop

Switched to a new branch "myfeature"

Sammanfogas tillbaka till develop.

$ git checkout develop

Switched to branch 'develop'

$ git merge --no-ff myfeature

Updating ea1b82a..05e9557

(Summary of changes)

$ git branch -d myfeature

Deleted branch myfeature (was 05e9557).

$ git push origin develop

### Anpassningar inför driftsättning

Från *develop.*

$ git checkout -b release-1.2 develop

Switched to a new branch "release-1.2"

$ ./bump-version.sh 1.2

Files modified successfully, version bumped to 1.2.

$ git commit -a -m "Bumped version number to 1.2"

[release-1.2 74d9424] Bumped version number to 1.2

1 files changed, 1 insertions(+), 1 deletions(-)

Sedan tillbaka till *master* och *develop*.

$ git checkout master

Switched to branch 'master'

$ git merge --no-ff release-1.2

Merge made by recursive.

(Summary of changes)

$ git tag -a 1.2

$ git branch -d release-1.2

Deleted branch release-1.2 (was ff452fe).

### Fixar

Kommer från *master:*

$ git checkout -b hotfix-1.2.1 master

Switched to a new branch "hotfix-1.2.1"

$ ./bump-version.sh 1.2.1

Files modified successfully, version bumped to 1.2.1.

$ git commit -a -m "Bumped version number to 1.2.1"

[hotfix-1.2.1 41e61bb] Bumped version number to 1.2.1

1 files changed, 1 insertions(+), 1 deletions(-)

Problemet fixas och fixen återförs till *master* och *develop*.

$ git commit -m "Fixed severe production problem"

[hotfix-1.2.1 abbe5d6] Fixed severe production problem

5 files changed, 32 insertions(+), 17 deletions(-)

$ git checkout master

Switched to branch 'master'

$ git merge --no-ff hotfix-1.2.1

Merge made by recursive.

(Summary of changes)

$ git tag -a 1.2.1

$ git checkout develop

Switched to branch 'develop'

$ git merge --no-ff hotfix-1.2.1

Merge made by recursive.

(Summary of changes)

$ git branch -d hotfix-1.2.1

Deleted branch hotfix-1.2.1 (was abbe5d6).

## Referenser

[Referens](https://git-scm.com/docs) till git dokumentation

## git commands

Clone a repository

```

git clone https://lars\_erik\_r\_jer\_s@bitbucket.org/lars\_erik\_r\_jer\_s/hippo.git dir

```

Switch to a different branch

```

git co branch-name

```

Create a new branch out of existing one and switch to it

```

git co -b new-branch existing-branch

```

Commit all changes into the branch

```

git ci -a -m "message"

```

Merge current branch to another branch

```

git merge --no-ff master-branch

```

Create a tag

```

git tag -a 1.2

```

Push a branch back to a master repository

```

git push origin branch-name

```

Delete a branch

```

git branch -d branch-name

```

# Appendix C – Versionsspecifika instruktioner

## TPINFO 5.2

### TPDB 5.2

Vid uppgradering till 5.2 från 5.1 behöver databasstrukturen förändras. Det sker huvudsakligen via SQL-scriptet Upgrade-5.1-to-5.2.sql. Dessutom får databasen ett nytt namn (skoview3) för att möjliggöra att köra den nya versionen parallellt med den gamla.

Följande skall följande göras i databasen.

1. Skapa användare med behörighet:
   1. sudo mysql
   2. GRANT ALL PRIVILEGES ON skoview2.\* TO 'skoview'@'localhost' IDENTIFIED BY 'xxxyyyzzz';
2. Importera den senaste backupen av 5.1-databasen. Själva db I dump-filen heter skoview2:
   1. mysql -u skoview -p
   2. use skoview2;
   3. source <filnamn.sql>
3. Kör uppdateringsscriptet. Det genomför ett antal förändringar av databasen, se scriptet för detaljer.:
   1. source Upgrade-5.1-to-5.2.sql;
4. Säkra att alla statistikfiler finns tillgängliga och gör en uppdatering. Detta kan också göras ett steg senare, först i skoview3-databasen.
5. Kopiera databasen (genom dump/restore)
   1. create database skoview3 DEFAULT CHARACTER SET utf8;
   2. GRANT ALL PRIVILEGES ON skoview3.\* TO 'skoview'@'localhost' IDENTIFIED BY 'xxxyyyzzz';
   3. mysqldump -u leo -p skoview2 > /tmp/skoview2.sql
   4. mysql -u leo -p skoview3 < /tmp/skoview2.sql
6. Verifiera att allt ser rätt ut
7. Flytta över skoview2.sql till produktionsserver och importera i mariadb pss som punkt 5.d. ovan. Observera att databasen i produktionsservern skall heta enbart ”skoview”, dvs utan siffra på slutet.

### Filstruktur på produktionsserver

Installationen måste möjliggöra att vi kan använda den nya versionen parallellt med den gamla, samt enkelt byta vilken som är default.

Historiskt så ligger koden installerad (egentligen bara uppackad) i mappen:

/var/www/html/skoview/

Här finns tre mappar där all html och kod ligger:

* lib/
* statdb/
* views/

Vid installation skall filerna i lib/ kopieras till ~/ini/. De har fått nya namn i 5.2, vilket gör att de kan peka ut en annan databas (skoview3).

Ovanstående mappar kopieras till:

/var/www/html/skoview/oldversion

Det går dock inte att köra den gamla versionen via länken till oldversion. Det finns alltför många hårdkodade referenser till URL-erna i koden för att det ska vara möjligt.

Den nya installationen av 5.2 sker sedan i:

/var/www/html/skoview/newversion

Koden i den nya versionen har delvis anpassats till denna URL. Det går att köra mot den. Observera att hanteringen av URL-er i logiken kan peka om dessa till en URL utan newversion i URL-en. Följande fyra fall blir fel:

* ”Statistik” knappen i hippo vid val av SLL-Prod. Den pekar på defaultadressen för statistiken.
* Länken till ”Till alla integrationer…” i Statistiken. Den perkar på defaultadressen för hippo.
* URL-omskrivningen i hippo pekar på defaultadressen.
* URL-omskrivningen i statistiken pekar på defaultadressen. Observera att det inte är möjligt att sätta rätt adress för statistiken pga säkerhetsspärrar i http-protokollet.

### cron-jobb

Jobbet för den gamla versionen uppdateras med rätt path till uppdateringsscriptet (dvs inkluderande ”oldversion”).

Ett jobb läggs till för den nya versionen.

* 30 5 \* \* \* cd /var/www/html/skoview/oldversion/statdb/ && php70 statdbupdate.php > /home/leoroj/logs/$(date --iso-8601)\_statdbupdate\_old.log
* 30 4 \* \* \* cd /var/www/html/skoview/newversion/statdb/ && php70 statdbupdate.php > /home/leoroj/logs/$(date --iso-8601)\_statdbupdate\_5.2.log

### Övergång från gammal till ny version

1. cd /var/www/html/skoview/
2. rm -rf statdb/
3. ln -s newversion/statdb
4. rm -rf views/
5. ln -s newversion/views

### Återgång till gamla versionen

1. cd /var/www/html/skoview/
2. rm -rf statdb/
3. ln -s oldversion/statdb
4. rm -rf views/
5. ln -s oldversion/views

### Uppstädning

När uppgraderingen är genomförd bör följande uppstädning ske:

* Tag bort alla mappar, filer och länkar från /var/www/html/skoview/ utom statdb/ och views/.
* Slå av cron-jobbet för den gamla miljlön.
* Flytta alla statistikfiler till arkivet.

## TPINFO 6.0

Med denna version flyttar vi in i den nya produktionsmiljön, och till docker.

### TPDB6.0

Innan en databas flyttas till produktionsservern skall de tre vyerna tas bort.

Därefter exporteras aktuell databas mha mysqldump. I detta exempel heter källdatabasen ”skoview3”. Exekvera följande kommandon för att exportera, komprimera och skicka dumpen till en produktionsserver:

mysqldump -u skoview -p --opt skoview3 > skoview3.sql

bzip2 skoview3.sql

scp skoview3.sql.bz2 larroj@tpinfo-a-web01.rtp.i.nogui.se:~/

I produktionsservern (tpinfo-a-web01 I detta fall) skall följande exekveras:

bzip2 -d skoview3.sql.bz2

mysql -u TPDB -p TPDB < skoview3.sql