Create table mx_project

```
create table if not exists mx_projects (
  pid serial PRIMARY KEY,
  id character varying(40) unique not null,
  id_old character varying(3),
  title jsonb,
  description jsonb,
  active boolean,
  public boolean default true,
  admins jsonb,
  members jsonb,
  publishers jsonb,
  map_position jsonb,
  countries jsonb,
  creator integer,
  date_created timestamp with time zone,
  date_modified timestamp with time zone
);

ALTER TABLE mx_projects OWNER TO mapxw;


CREATE INDEX ON mx_projects (id);
CREATE INDEX ON mx_projects (id_old);
CREATE INDEX ON mx_projects USING gin (members);
CREATE INDEX ON mx_projects USING gin (publishers);
CREATE INDEX ON mx_projects USING gin (admins);
```

Convert views and sources columns

views

```
ALTER TABLE mx_views ADD COLUMN project varchar(22);
ALTER TABLE mx_views ALTER project TYPE varchar(22) USING country;
UPDATE mx_views SET data = jsonb_set(data,'{"projects"}',data -> 'countries',true) WHERE dat
/* delete old key*/
UPDATE mx_views SET data = data - 'countries'  WHERE data -> 'countries' IS NOT NULL;
ALTER TABLE mx_views ADD COLUMN readers jsonb;
ALTER TABLE mx_views ADD COLUMN editors jsonb;
CREATE INDEX ON mx_views USING gin (readers);
CREATE INDEX ON mx_views USING gin (editors);
```

sources

```
ALTER TABLE mx_sources ADD COLUMN project varchar(22);
ALTER TABLE mx_sources ALTER project type varchar(22) using country;
ALTER TABLE mx_sources ADD COLUMN readers jsonb;
ALTER TABLE mx_sources ADD COLUMN editors jsonb;
CREATE INDEX ON mx_sources USING gin (readers);
CREATE INDEX ON mx_sources USING gin (editors);
```

Update views readers and editors

views

```
UPDATE mx_views SET readers =
 CASE WHEN target@>'["public"]' THEN '["public"]'::jsonb ELSE '[]' END ||
 CASE WHEN target@>'["publisher"]' OR target@>'["public"]' THEN '["publishers"]'::jsonb ELSE
 CASE WHEN target@>'["user"]' THEN '["members"]'::jsonb ELSE '[]' END ||
 CASE WHEN target@>'["admin"]' OR target@>'["superuser"]' THEN '["admins"]'::jsonb ELSE '[]'

UPDATE mx_views SET editors =
 CASE WHEN target@>'["publisher"]' OR target@>'["public"]' THEN '["publishers"]'::jsonb ELSE
 CASE WHEN target@>'["admin"]' OR target@>'["superuser"]' THEN '["admins"]'::jsonb ELSE '[]'
```

Sources

```
UPDATE mx_sources SET readers =
 CASE WHEN target@>'["public"]' THEN '["public"]'::jsonb ELSE '[]' END ||
 CASE WHEN target@>'["publisher"]' OR target@>'["public"]' THEN '["publishers"]'::jsonb ELSE
 CASE WHEN target@>'["user"]' THEN '["members"]'::jsonb ELSE '[]' END ||
 CASE WHEN target@>'["admin"]' OR target@>'["superuser"]' THEN '["admins"]'::jsonb ELSE '[]'

UPDATE mx_sources SET editors =
 CASE WHEN target@>'["publisher"]' OR target@>'["public"]' THEN '["publishers"]'::jsonb ELSE
 CASE WHEN target@>'["admin"]' OR target@>'["superuser"]' THEN '["admins"]'::jsonb ELSE '[]'
```

Convert country table to project table

```r
source('global.R')
countries <- config$countries$table
countries_iso3 <- countries[!countries$iso3 %in% c("XXX"),c("iso3")]
languages <- as.character(config$languages$list)
dictCountries <- config$dictionaries$countries

titles <- lapply(countries_iso3,function(c){

  res <- lapply(languages,function(l){
    d(c,l,web=F,dict=dictCountries)
})
  names(res)<-languages
  res <- toJSON(res,auto_unbox=T)
  res <- as.character(res)
  return(res)
})

mapPos <- lapply(countries_iso3,function(c){
  pos <- as.list(countries[countries$iso3 == c,c("lng","lat","zoom")][1,])
  pos <- toJSON(pos,auto_unbox=T)
  return(pos)
})


for(i in 1:length(countries_iso3)){

r <- list(
    pid = i,
    id = randomString("MX-",splitIn=5,addLetters=F,addLETTERS=T,splitSep="-",sep="-"),
    id_old = countries_iso3[i],
    title = titles[[i]],
    description = list(),
    map_position = mapPos[[i]],
    active = TRUE,
    creator = 1,
    admins = toJSON(1),
    members = toJSON(list()),
    publishers = toJSON(1),
    countries = toJSON(countries_iso3[i]),
    date_modified = Sys.time(),
    date_created = Sys.time(),
    public = TRUE
    )
  mxDbAddRow(r,"mx_projects")

}
#
# Set default project with custom id
#
mxDbGetQuery("update mx_projects set id ='MX-YHJ-6JJ-YLS-SCV-VL1' where id_old ='COD'");
```

Update views based on project

Update views

```
WITH ids as (
    select distinct id, id_old from mx_projects
)
UPDATE mx_views SET
project = ids.id
FROM ids
WHERE mx_views.country = ids.id_old  ;
```

Update source

```
WITH ids as (
    select distinct id, id_old from mx_projects
)
UPDATE mx_sources SET
project = ids.id
FROM ids
WHERE mx_sources.country = ids.id_old;
```

Update user last project

```
WITH ids as (
    select distinct to_jsonb(id) id, to_jsonb(id_old) id_old from mx_projects
)
UPDATE mx_users SET
data = jsonb_set(data,'{"user","cache","last_project"}',ids.id,true)
```

```
FROM ids
WHERE mx_users.data #> '{"user","cache","last_project"}' = ids.id_old;
```

Update shared view

```
WITH expended as (
 select id, jsonb_array_elements_text(data->'projects') as project_old
 FROM mx_views
 WHERE jsonb_typeof(data->'projects') = 'array'
),
corrected as (
SELECT p.id as project, v.id from mx_projects p, expended v where p.id_old = v.project_old
),
aggregated as (
  select jsonb_agg(project) project, id from corrected group by id
)
update mx_views set
data = jsonb_set(data,'{projects}',aggregated.project,true)
from aggregated
where mx_views.id = aggregated.id;
```

Set publishers based on people that already have published something in project

```
WITH pub as (
    select jsonb_agg(distinct(editor)) ids, project from mx_views group by project
)
UPDATE mx_projects SET
publishers = pub.ids
FROM pub
WHERE mx_projects.id = pub.project;
```

Set default admins

```
UPDATE mx_projects SET
admins = '[1,17]'::jsonb;
```

Remove countries columns

```
alter table mx_views drop column country;
alter table mx_sources drop column country;
```

Create postgres view version of latest mapx view

```
CREATE INDEX mx_views_id_latest_idx on mx_views (id, date_modified DESC NULLS LAST);

CREATE or REPLACE view mx_views_latest as (
    WITH latest_date as (
      SELECT id, max(date_modified) date_latest
      FROM mx_views
      GROUP by id
      ),
    subviews as (
      SELECT pid
      FROM mx_views, latest_date
      WHERE mx_views.id = latest_date.id
      AND date_modified = latest_date.date_latest
      )
    SELECT mx_views.*
    FROM mx_views,subviews
    WHERE mx_views.pid = subviews.pid
    );
```

For the temporary user, mapxw need to be able to create role

```
alter user mapxw with createrole;
```

Update titles (solve bug where titles where missing)

```
source('global.R')
countries <- config$countries$table
  countries_iso3 <- countries[!countries$iso3 %in% c("XXX"),c("iso3")]
languages <- as.character(config$languages$list)
  dictCountries <- config$dictionaries$countries
  titles <- lapply(countries_iso3,function(c){

      res <- lapply(languages,function(l){
          d(c,l,web=F,dict=dictCountries)
          })
      names(res)<-languages
      res <- toJSON(res,auto_unbox=T)
      res <- as.character(res)
      return(res)
      })


for(i in 1:length(countries_iso3)){

  mxDbUpdate(
      table = "mx_projects",
      id = countries_iso3[i],
      idCol = "id_old",
      column = "title",
      value = titles[[i]]
      )

}
```