**В статье опишу порядок установки настройки и использования PostgREST на примере организации каталога товаров.**

**Вводные**

* Торговая компания работающая в сфере B2B
* Нужно опубликовать каталог товаров для автоматической загрузки данных из него в базы клиентов
  + Группы номенклатуры
  + Номенклатура
  + Цены
  + Остатки

На мысль о данной реализации навела статья [//infostart.ru/1c/articles/1217831/](https://infostart.ru/1c/articles/1217831/), но в ней больше описано про подход и реализацию со стороны 1С. В своей статье опишу настройку сервиса, формирование БД, настройку доступа и обращение по HTTP.

**Вариант реализации**

Установка сервиса PostgREST

Разворачивать сервис буду на VDS с Ubuntu 20.04, хостинг digitalocean.

Подключаюсь к серверу по SSH с помощью PuTTY ([ссылка](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cuY2hpYXJrLmdyZWVuZW5kLm9yZy51ay9+c2d0YXRoYW0vcHV0dHkvbGF0ZXN0Lmh0bWw=)) используя SSL ключ ([ссылка](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cuZGlnaXRhbG9jZWFuLmNvbS9kb2NzL2Ryb3BsZXRzL2hvdy10by9hZGQtc3NoLWtleXMvY3JlYXRlLXdpdGgtcHV0dHkv)). Ключ можно указать в настройках подключения, так и загрузить в Pageant из группы программ PuTTY. Ключ в дальнейшем буду использовать для подключения к PostgreSQL через SSH туннель.

**Добавляю разрешения в брандмауэр, включаю его**

#Добавляю разрешения

sudo ufw allow ssh

sudo ufw allow http

sudo ufw allow https

#Включаю брандмауэр

sudo ufw enable

Более подробно по [ссылке](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cuZGlnaXRhbG9jZWFuLmNvbS9jb21tdW5pdHkvdHV0b3JpYWxzL2hvdy10by1zZXQtdXAtYS1maXJld2FsbC13aXRoLXVmdy1vbi11YnVudHUtMTgtMDQtcnU=)

Для работы потребуется PostgreSQL (PG), для Ubuntu 20.04, PostgREST поддерживает версию 9.4 и выше, в стандартных репозиториях ubuntu 12 версия. Инструкция по установке [https://www.digitalocean.com/community/tutorials/how-to-install-postgresql-on-ubuntu-20-04-quickstart-ru](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cuZGlnaXRhbG9jZWFuLmNvbS9jb21tdW5pdHkvdHV0b3JpYWxzL2hvdy10by1pbnN0YWxsLXBvc3RncmVzcWwtb24tdWJ1bnR1LTIwLTA0LXF1aWNrc3RhcnQtcnU=)

# Устанавливаем postgres

sudo apt install postgresql postgresql-contrib

**Устанавливаю пароль пользователя, добавляю схему**

# При установке будет создан пользователь postgres, он состыкован с одноименным пользователем в PostgreSQL

# Переходим в сеанс пользователя postgres

sudo -i -u postgres

#Запускаю psql

psql

# Устанавливаю пароль пользователя postgres, под ним буду подключаться к базе через клиента

ALTER USER postgres WITH PASSWORD 'NojmyXBOiOnoDacmuecd';

# Добавляю пользователя для подключения из сервиса

CREATE USER authenticator NOINHERIT LOGIN PASSWORD 'NojmyXBOiOnoDacmuecd';

#Создаю схему api

create schema api;

#Добавляю роль анонимного пользователя

create role web\_anon nologin;

#Наследую права доступа web\_anon для authenticator

grant web\_anon to authenticator;

#Разрешаю использовать схему api

grant usage on schema api to web\_anon;

#Выхожу из psql

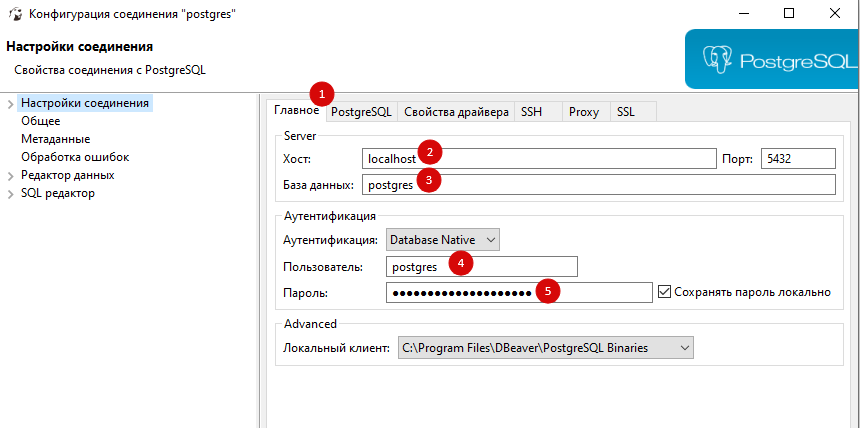
\q

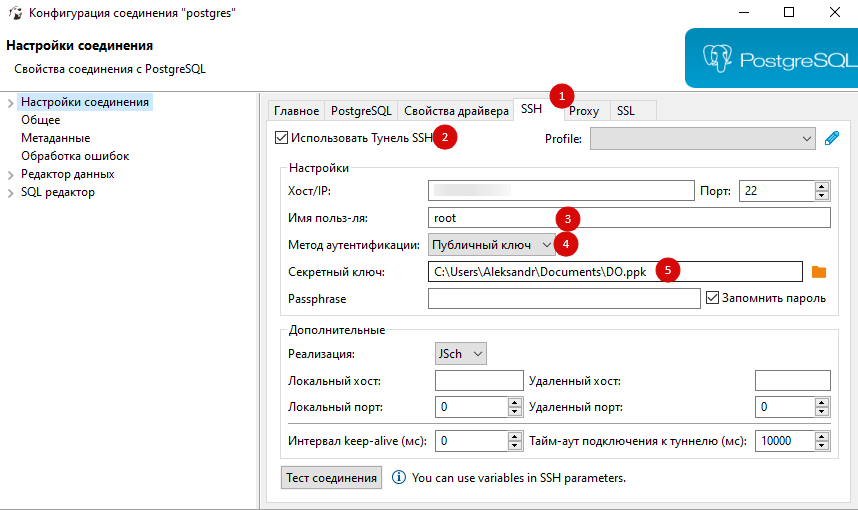
#Выхожу из пользователя postgres

exit

Для подключения к базе PG с рабочего места использую DBeaver [https://dbeaver.io/](https://infostart.ru/redirect.php?url=aHR0cHM6Ly9kYmVhdmVyLmlvLw==), ([ссылка на видео по функционалу](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1BMmU2SlVXSlBzNA==)). При подключении использую SSH туннель с SSL ключом.

**Скриншоты настройки подключения**





Сервис будет работать на PostgREST http://postgrest.org/en/v7.0.0/

Для работы сервиса PostgREST буду использовать отдельного пользователя с ограниченными правами ([ссылка на статью по теме](https://infostart.ru/redirect.php?url=aHR0cHM6Ly93d3cuZGlnaXRhbG9jZWFuLmNvbS9jb21tdW5pdHkvdHV0b3JpYWxzL2hvdy10by1jcmVhdGUtYS1uZXctc3Vkby1lbmFibGVkLXVzZXItb24tdWJ1bnR1LTIwLTA0LXF1aWNrc3RhcnQ=)).

# Добавляю пользователя ubuntu

adduser pgadmin

# указываю пароль и остальные параметры

Приложение PostgREST поставляется в виде бинарного файла и запускается с параметрами подключения, которые можно зафиксировать в конфигурационном файле, пример файла можно получить запустив ./postgres --help

Получаем ссылку на архив по ссылке [https://github.com/PostgREST/postgrest/releases/latest](https://infostart.ru/redirect.php?url=aHR0cHM6Ly9naXRodWIuY29tL1Bvc3RnUkVTVC9wb3N0Z3Jlc3QvcmVsZWFzZXMvbGF0ZXN0) ([ссылка на документацию](https://infostart.ru/redirect.php?url=aHR0cDovL3Bvc3RncmVzdC5vcmcvZW4vdjcuMC4wL3R1dG9yaWFscy90dXQwLmh0bWw=))

**UPD:** Если используете PG14, то нужная версия v9.0.0 и выше

**Получение приложения, настройка**

#переходим в папку пользователя pg\_admin

cd /home/pgadmin/

#Скачиваем архив

wget https://github.com/PostgREST/postgrest/releases/download/v7.0.1/postgrest-v7.0.1-linux-x64-static.tar.xz

# Распаковываем архив

tar -xf postgrest-v7.0.1-linux-x64-static.tar.xz

# Добавляем файл

nano postgrest.conf

# Содержимое файла

db-uri = "postgres://authenticator:NojmyXBOiOnoDacmuecd@localhost:5432/postgres"

db-schema = "api"

# this schema gets added to the search\_path of every request

db-anon-role = "web\_anon"

db-pool = 10

db-pool-timeout = 10

server-host = "!4"

server-port = 3000

#меняем владельца файлов на pgadmin

#загружали файлы из под root, без этого будет ошибка доступа к файлам

chown -R pgadmin:pgadmin /home/pgadmin/

Создаем, запускаем службу ([ссылка](https://infostart.ru/redirect.php?url=aHR0cDovL3Bvc3RncmVzdC5vcmcvZW4vdjcuMC4wL2FkbWluLmh0bWw=), раздел Daemonizing)

**Настройка службы linux**

#создаю файл сервиса

nano /etc/systemd/system/postgrest.service

#Содержимое файла

[Unit]

Description=REST API for any Postgres database

After=postgresql.service

[Service]

Type=simple

User=pgadmin

ExecStart=/home/pgadmin/postgrest /home/pgadmin/postgrest.conf

ExecReload=/bin/kill -SIGUSR1 $MAINPID

[Install]

WantedBy=multi-user.target

#Обновляю список сервисов

systemctl daemon-reload

#Запускаю сервис

systemctl start postgrest

#Проверяю корректность запуска

systemctl status postgrest

#Устанавливаю атозапуск

systemctl enable postgrest

Служба работает на 3000 порту, опубликуем её на 80 используя nginx

**Установка, настройка nginx**

#Устанавливаю nginx

sudo apt install nginx -y

#создаю файл

sudo nano /etc/nginx/sites-available/postgrest

# Содержимое

server {

listen 80;

server\_name postgrest.malikov.pro;

#reverce proxy

location / {

proxy\_pass http://localhost:3000;

}

}

# После сохранения файла

# создаю ссылку для активации настроек

sudo ln -s /etc/nginx/sites-available/postgrest /etc/nginx/sites-enabled/

# проверяю корректность настроек

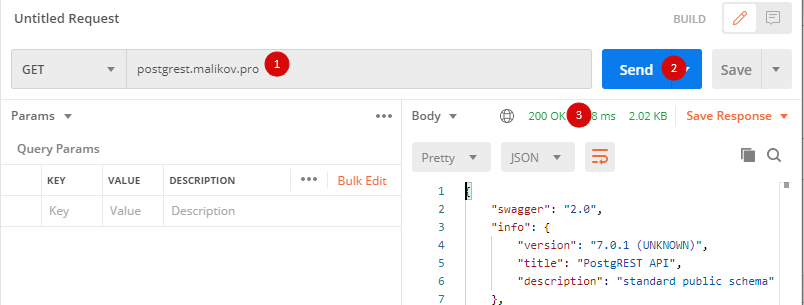
sudo nginx-t

# загружаю изменения настроек в сервис

sudo service nginx reload

Проверяем подключение.

**скриншот**



Создание структуры базы данных

**Запрос создания таблицы**

create table api.products (

id serial primary key,

name text NOT NULL,

unit\_name text NOT NULL,

price numeric,

count numeric,

created\_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

updated\_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),

completed\_at TIMESTAMPTZ

);

При изменении строк фиксирую дату изменения, это реализовано через триггеры ([документация](https://infostart.ru/redirect.php?url=aHR0cHM6Ly9wb3N0Z3Jlc3Byby5ydS9kb2NzL3Bvc3RncmVzcHJvLzEyL3BscGdzcWwtdHJpZ2dlcg==)) ([ссылка на статью по теме](https://infostart.ru/redirect.php?url=aHR0cHM6Ly94LXRlYW0uY29tL2Jsb2cvYXV0b21hdGljLXRpbWVzdGFtcHMtd2l0aC1wb3N0Z3Jlc3FsLw==))

**Запросы создания функции и подключения к таблице через триггеры**

 Создание функции

CREATE OR REPLACE FUNCTION api.trigger\_set\_timestamp()

RETURNS TRIGGER AS $$

BEGIN

NEW.updated\_at = NOW();

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

Добавление триггера

CREATE TRIGGER set\_timestamp

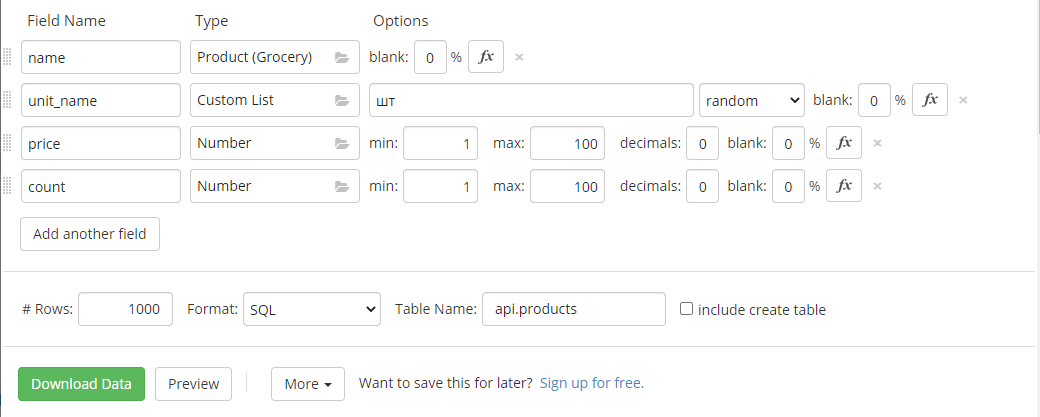
BEFORE UPDATE ON api.products

FOR EACH ROW

EXECUTE PROCEDURE api.trigger\_set\_timestamp();

Заполняю тестовыми данными, для этого использую https://www.mockaroo.com/

**Скриншот настройки генерации**



Полученный файл запускаю через DBeaver.

Настройка прав доступа

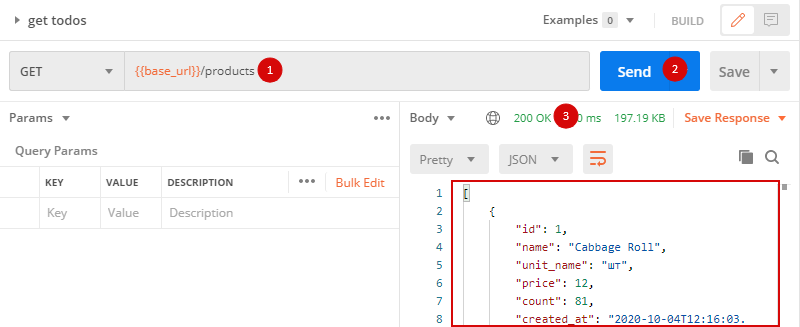
([Ссылка на урок](https://infostart.ru/redirect.php?url=aHR0cDovL3Bvc3RncmVzdC5vcmcvZW4vdjcuMC4wL3R1dG9yaWFscy90dXQxLmh0bWw=))

Добавляю права доступа для web\_anon на чтение таблицы api.products

grant select on api.products to web\_anon;

Проверяю результат

**Скриншот HTTP запроса**



Добавляю пользователя с правом записи

**SQL команды**

#Добавляю пользователя

create role admin\_user nologin;

#Расширяю права пользователя authenticator

grant admin\_user to authenticator;

#Разрешаю доступ к схеме

grant usage on schema api to admin\_user;

#Разрешаю полный доступ к базе

grant all on api.products to admin\_user;

#Разрешаю доступ к последовательности таблицы (используется для формирования идентификатора записи)

grant usage, select on sequence api.products\_id\_seq to admin\_user;

Для аутентификации на уровне PostgREST используется JWT формата HS256 ([статья по работе из 1С](https://infostart.ru/1c/articles/611505/)) , ключ фиксируется в конфигурационном файле.

**Добавление настройки JWT ключа**

# Открываем файл на редактирование

nano /home/pgadmin/postgrest.conf

# Добавляем строку настройки, значение указываем свое

# если используете спецсимволы, то нужно сохранять в Base64 и указывать соответствующий флаг

jwt-secret = "SWFCI2oJCo52f2vhEZ7RoJk5fLsH9qj5"

# Перезагружаем сервис

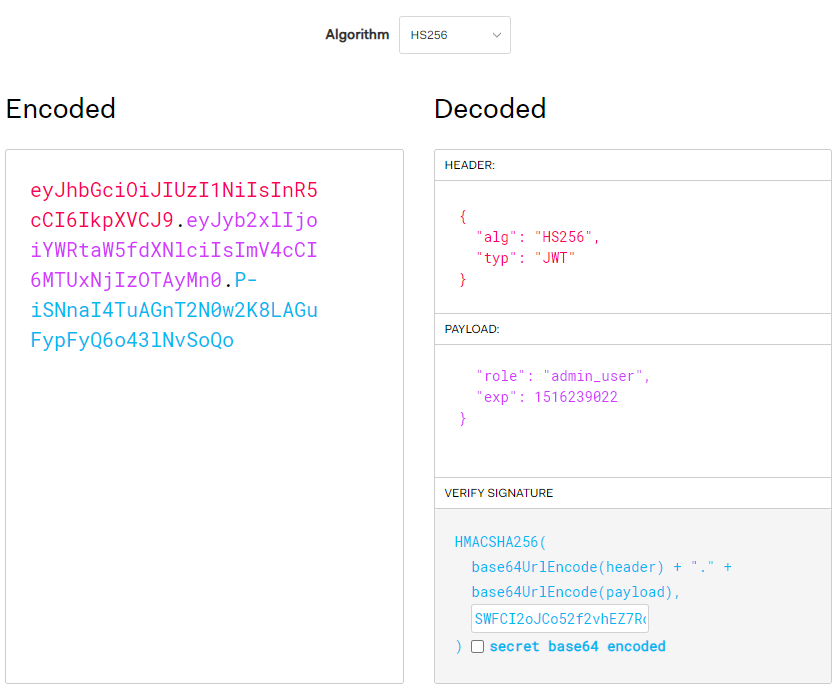
systemctl restart postgrest

Токен можно сгенерировать используя сервис https://jwt.io/, результат помещается в заголовок HTTP запроса

"Authorization: Bearer $JWT\_TOKEN"

В теле запроса на получение токена указывается поле "role", для которой выдается токен. При таком подходе токен будет бессрочным, что затрудняет его отзыв. Для решения вопроса можно добавить поле "exp" указывающее на время истечения токена.

**Скриншот с примером генерации ключа**



Для большей гибкости можно использовать параметры из тела JWT токена, пример ограничения доступа по email

**Команды SQL**

#Добавялем схему auth

create schema auth;

#Разрешаем использовать схему пользователям,

#брал из оф. примеров, по идее можно указать "to authenticator"

grant usage on schema auth to web\_anon, admin\_user;

#Добавляем функцию проверяющую email

# request.jwt.claim берется из текущих настроек запроса

create or replace function auth.check\_token() returns void

language plpgsql

as $$

begin

if current\_setting('request.jwt.claim.email', true) =

'disgruntled@mycompany.com' then

raise insufficient\_privilege

using hint = 'Nope, we are on to you';

end if;

end

$$;

**Добавляем использование функции в настройки сервиса**

# Открываем файл на редактирование

nano /home/pgadmin/postgrest.conf

# Добавляем строку настройки

pre-request = "auth.check\_token"

# Перезагружаем сервис

systemctl restart postgrest

При формировании токена в тело добавляем параметр email

{

"role": "todo\_user",

"email": "disgruntled@mycompany.com"

}

Эти данные можно использовать в представлениях для отбора, например цены по доступным для пользователя с email.

Добавление, обновление данных

([Ссылка на документацию](https://infostart.ru/redirect.php?url=aHR0cDovL3Bvc3RncmVzdC5vcmcvZW4vdjcuMC4wL2FwaS5odG1sI2luc2VydGlvbnMtdXBkYXRlcw==))

Для добавления данных используется POST запрос с JSON в теле запроса, поддерживается групповая загрузка, пример

[

{"name":"Chicken - Leg, Fresh","unit\_name":"шт","price":57,"count":97},

{"name":"Chicken Giblets","unit\_name":"шт","price":100,"count":25}

]

Для обновления данных используется PATCH, в параметрах запроса указывается идентификатор, например "?id=eq.1002" в теле обновляемые поля, например

{

"price":10020

}

**Итог**

С помощью PostgREST можно организовать сервис работы с данными использующий функционал Postgres с удобной оберткой в HTTP.

Благодарю за внимание.