**PostgreSQL**

Установка posgresql:

pip install psycopg2-binary

brew install postgres

pg\_ctl -D /usr/local/var/postgres start | stop | restart | status

psql postgres

Список баз данных:

psql -l

Либо:  
select \* from pg\_database;

# \l

Список пользователей:

select \* from pg\_shadow;

# \du

Создание базы данных:

CREATE DATABASE sysadmin;

Далее вам необходимо создать нового пользователя базы данных (sysadminuser) для того, чтобы подключиться к базе данных и взаимодействовать с ней. Обязательно установите длинный и сложный пароль:

CREATE USER sysadminuser WITH PASSWORD '1234';

Установите UTF-8 в качестве кодировки по умолчанию (это необходимо для Django). Уровень изолированности транзакций нужно по умолчанию установить как “read committed” (чтение фиксированных данных) – этот уровень обеспечивает защиту от чернового чтения. Не забудьте про временную зону: изначально проекты на Django имеют установку UTC. Чтобы выполнить все изменения, нужно ввести следующие команды:

ALTER ROLE sysadminuser SET client\_encoding TO 'utf8';

ALTER ROLE sysadminuser SET default\_transaction\_isolation TO 'read committed';

ALTER ROLE sysadminuser SET timezone TO 'UTC';

Теперь последнее, что необходимо сделать, это дать только что созданному пользователю права доступа к созданной базе:

GRANT ALL PRIVILEGES ON DATABASE sysadmin TO sysadminuser;

Заходим в наш Django проект и в файле settings.py меняем sqlite3 на postgresql:

DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql'**,  
 **'NAME'**: **'sysadmin'**,  
 **'USER'**: **'sysadminuser'**,  
 **'PASSWORD'**: **'1234'**,  
 **'HOST'**: **'localhost'**,  
 **'PORT'**: **''**,  
 }  
}

Выполняем минрации:

python manage.py makemigrations

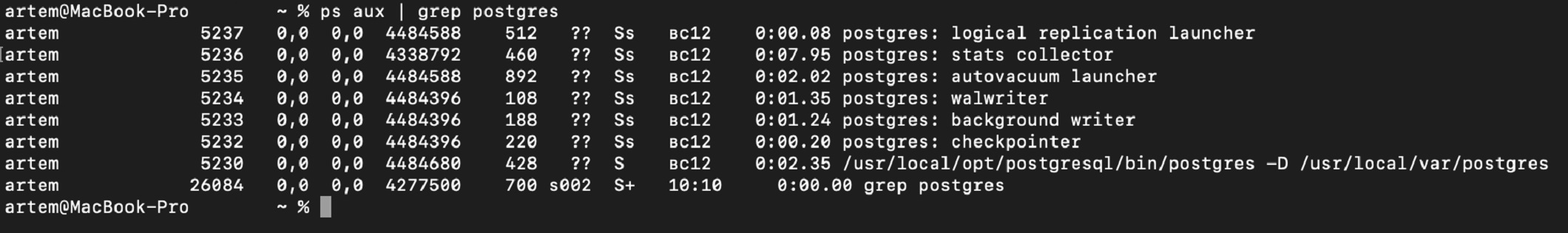
python manage.py migrate

Подключение к базе данных databasename

\c sysadmin;

**Конфиги postgresql**

ps aux | grep postgres



В каталоге /usr/local/var/postgres находим файл pg\_hba.conf

**Отключить подключение к базе данных без пароля при локальном подключении**

В файле pg\_hba.conf в строчке ‘IPV6 local connections:’ устанавливаем md5



Для подключения к postgres пишем в консоли:

psql postgres

Список баз данных:

\l

Подключение к базе данных:

\connect sysadmin

Получение списка имеющихся таблиц:

\dt

Просмотр содержимого таблицы:

SELECT \*

FROM account\_profile;

Удаление из таблицы:

DELETE FROM django\_migrations

WHERE app='account';

получить список имен столбцов и типов данных таблицы

\d+ blog\_post

1. sudo apt install postgresql

2. python -m pip install psycopg2-binary

3. sudo su - postgres

4. psql

5. create role djangouser with login password '1234';

6. create database namedb with owner djangouser;

7. \q

8. exit

9. psql namedb -h localhost -U djangouser

10. Изменить настройки в settings.py:

DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql\_psycopg2'**,  
 **'NAME'**: **'inportal'**,  
 **'USER'**: **'djangouser'**,  
 **'PASSWORD'**: os.getenv(**'SECRET\_KEY\_DB'**),  
 **'HOST'**: **'localhost'**,  
 **'PORT'**: **'5432'**,  
 }  
}

11. python manage.py migrate

**Установка пароля для postgresql**

userpc@userpc:~$ sudo su — postgres

postgres@userpc:~$ psql

postgres-# \password postgres

Enter new password:

Enter it again:

**Резервное копирование базы данных**

pg\_basebackup -h localhost -U postgres -D ./BackUp/

-U, --username=NAME подключение в качестве указанного пользователя базы данных

-D, --pgdata=DIRECTORY получить базовую резервную копию в каталог

Создаем бэкап с помощью pg\_dump:

userpc@userpc:~$ pg\_dump -U postgres -W -h localhost insys\_test > ./BackUp/outfile

Перед восстановлением создаем базу данных «insys\_test»:

postgres=# create database insys\_test with owner djangouser;

Для восстановления бэкапа выполняем:

userpc@userpc:~$ psql -U postgres -W -h localhost insys\_test < ./BackUp/outfile