**Установка сервиса**

в PGSQL создать БД, например, filestorage выполнить

CREATE TABLE IF NOT EXISTS files (guid uuid PRIMARY KEY,

name varchar (150) NOT NULL,

extension varchar (10) NOT NULL,

description text,

date\_added date DEFAULT CURRENT\_TIMESTAMP);

далее

pip install virtualenv

выполняем в папке , где будет сервис:

$ mkdir fileservice-api

$ cd fileservice-api

$ virtualenv flask

$ flask/bin/pip install flask

$ flask/bin/pip install psycopg2-binary

$ flask/bin/pip install waitress

$ flask/bin/pip install cryptography

положить в fileservice-api содержимое архива

сгенерировать ключ криптографии:

$ chmod a+x create-key.py

$ ./create-key.py

в результате в этой папке появится файл crypto.key

поместить в папку, куда доступ будет иметь ограниченный круг лиц. ВНИМАНИЕ! без данного ключа расшифровка файлов будет невозможна!!

прописать в app.py параметры:

cryptokeypath = '/home/sngadmin/fileservice-api/crypto.key' - полный путь где лежит сгенерированный ключ

dirpath = '/opt/files/' - где будут файлы храниться

dirpathtmp = '/opt/files/tmp/' - где будут временные фалйы

httpfileserver = 'localhost:8081' - вместо localhost адрес сервака

pg\_server = "localhost" - адрес сервера СУБД

pg\_user = "postgres" - юзер PGSQL

pg\_password = "Zraeqw123" - пароль юзера PGSQL

pg\_database = "filestorage" - имя БД

при необходимости поменять порты на нужные в строках

httpfileserver = 'localhost:8081'

serve(app, host="0.0.0.0", port=8081)

выполнить

$ chmod a+x app.py

запустить для теста

$ ./app.py

проверить ,что работает зайти на localhost:8081 - будет приветственная страница

$ ctrl+C

далее изменяем файл fileservice\_galaktika.service

меняем строки:

на пользователя под которым будет выполняться сервис

User=sngadmin

меняем пути на актуальные

ExecStart=/home/sngadmin/fileservice-api/flask/bin/python3 /home/sngadmin/fileservice-api/app.py

копируем измененный fileservice\_galaktika.service в /etc/systemd/system/

далее

# sudo systemctl daemon-reload

# sudo systemctl start fileservice\_galaktika

# sudo systemctl status fileservice\_galaktika

если статус active(running), то делаем автозапуск сервиса:

# sudo systemctl enable fileservice\_galaktika

**Описание методов**

def load\_key():

Загружаем ключ 'crypto.key' из каталога cryptokeypath (указан в параметрах)

Шифрование идет с помощью библиотеки Fernet

def encrypt(filename, key):

Зашифруем файл и записываем его в папку-хранилище

Шифрование идет с помощью библиотеки Fernet

def decrypt(filename, key):

Расшифруем файл и записываем его во временную папку

def cleartempdir():

очищаем все файлы из временной папки старше 15 минут

def get\_db\_connection():

получить подключение к БД

@app.errorhandler(404)

def not\_found(error):

функция API - возвращает ошибку 404 при обращении к несуществующей странице

@app.route('/fileservice/api/v1.0/loadfile', methods=['POST'])

def load\_file():

функция API - по загрузке файла в сервис.

Входные проверки

- если структура входного JSON некорректная то возвращает

{'message': 'Wrong JSON structure in body', 'status': '0', 'UUID': ''}, 401

- если название файла пустое, то возвращает

{'message': 'Empty filename', 'status': '0', 'UUID': ''}, 401

- если передат пустой файл, то возвращает

{'message': 'Empty data in file', 'status': '0', 'UUID': ''}, 401

Далее пытаемся записать файл на диск в директорию-хранилище, имя файлу присваивается уникальное (GUID) без расширения

-Если запись не удалась , то возврашщает

{'message': 'Errror write data to disk', 'status': '0', 'UUID': ''}, 401

Далее записанный файл зашифровываем функцией encrypt

- Если зашифровать и перезаписать файл не удалось, то возвращает

{'message': 'Errror encrypt file on disk', 'status': '0', 'UUID': ''}, 401

Далее записываем в БД имя файла на диске (сгенерированный GUID), изначальное имя, расширение файла и его описание

- если запись в БД не удалась, то возвращает

{'message': 'DB exception','status': '0','UUID':''}, 401

Если все прошло успешно, то возвращает GUID файла

{'message': 'OK','status': '1','UUID': guid}, 200

@app.route('/fileservice/api/v1.0/getfile/<file\_uuid>', methods=['GET'])

def get\_file(file\_uuid):

получение файла по его GUID

-Если передан пустой GUID, то возвращает

{'error': 'empty guid'}, 401

Далее ищем в БД файл по GUID

- Если ошибка запроса в БД, то возвращает

{'status': '0','message':'DB Exception','URL':''}, 401

- Если по GUID не нашли в БД запись, то возвращает

{'status': '0','message':'file not exists','url':''} , 401

Далее копируем файл во временную папку, переименовывая его str(uuid.uuid4())+'\_'+file\_name+'.'+file\_ext (file\_name, file\_ext берутся из БД) и расшифровываем его функцией decrypt

- Если не удалось скопировать, то возвращает

{'status': '0','message':'OS IO error','URL':''}, 401

Если все прошло хорошо, то возвращаем временную ссылку на файл

{'status': '1','message':'','url':httpfileserver+ '/fileservice/api/v1.0/tmpfiles/'+tmp\_file\_name}, 200

@app.route("/fileservice/api/v1.0/tmpfiles/<path>")

def DownloadFile (path = None):

функция скачивает файл по ссылке

Если файл не найден, то возвращает

return {'error': 'File not exists'}

@app.route('/fileservice/api/v1.0/deletefile/<file\_uuid>', methods=['DELETE'])

def delete\_file(file\_uuid):

Удаляет файл в хранилище и в БД

Делаем запрос на удаление записи в БД по GUID,

- если ошибка запроса, то возвращает

{'status': '0','message':'DB exception'},401 #error #{'error': 'DB exception'}

- если запись по GUID не найдена, то возвращает

{'status': '0','message':'File not exists'},401

Удаляем фйл на диске.

- Если удалить не удалось, то возвращает

{'status': '0', 'message': 'OS IO error'}, 401

- Если все прошло хорошо, то возвращает

{'status': '1','message':''},200

@app.route('/')

def index():

показывает страницу сервиса в веб-браузере