Тестовое задание

Необходимо разработать web-сервис по закачке/скачке данных.

В БД есть таблица со списком зарегистрированных пользователей.

Пользователь передаёт на сервер логин/пароль и получает авторизационный токен (session-id) для доступа к API.

```
$ curl -d '{"login":"alice","password":"secret"} http://myhost.com/api/auth

{"token":"2bdbbb11806cd18a90d730e61fbb54b5"}
```

Используя токен, пользователь может закачивать данные на сервер:

```
$ curl -X POST -H 'Authorization: Bearer 2bdbbb11806cd18a90d730e61fbb54b5' -d
'Hello, Alice!' http://myhost.com/api/upload-asset/hello

{"status":"ok"}
```

И скачивать данные с сервера:

```
$ curl -H 'Authorization: Bearer 2bdbbb11806cd18a90d730e61fbb54b5'
http://myhost.com/api/asset/hello

Hello, Alice!
```

Для реализации микросервиса можно пользоваться только стандартной библиотекой Go (последней версии) и библиотекой pgx для доступа к БД.

HTTP REST API Specs

Стандартные коды ошибок (HTTP статусы)

200 OK - порядок

400 Bad Request - Некорректный запрос. Например, отсутствует необходимое поле или запрос имеет не валидный JSON формат.

401 Unauthorized - Отсутствует авторизационный токен. Авторизационный токен не валидный/просроченный.

403 Forbidden - Доступ запрещен. Попытка обратиться к ресурсу, принадлежащему другому пользователю.

404 Not Found - Запрашиваемый ресурс не найден.

500 Internal Server Error - Ошибка сервера. Какая-то непредвиденная ошибка. Например БД недоступна.

```
{"error":"description"}
```

Сервис всегда возвращает данные в формате JSON, кроме случаев, когда явно необходим другой формат данных.

POST /api/auth

Request:

```
{
 "login": "myname",
 "password": "secret123"
}
```

Response:

```
{"token":"2bdbbb11806cd18a90d730e61fbb54b5"}
```

401 Unauthorized - не правильный логин/пароль

```
{"error": "invalid login/password"}
```

POST /api/upload-asset/{asset-name}

Request:

Body: any-data

Response:

```
{"status":"ok"}
```

GET /api/asset/{asset-name}

Response:

```
asset-body
```

---

schema.sql

```
create extension if not exists pgcrypto;

-- таблица с пользователями
create table if not exists users (
  id            bigserial primary key,
  login         text not null unique,
  password_hash text not null,
  created_at    timestamptz not null default now()
);

-- таблица сессий
create table if not exists sessions (
  id         text primary key default encode(gen_random_bytes(16),'hex'),
  uid        bigint not null,       -- user id
  created_at timestamptz not null default now()
);

-- таблица с файлами
create table if not exists assets (
  name       text not null,
  uid        bigint not null,       -- user id
  data       bytea not null,
  created_at timestamptz not null default now(),
  primary key (name, uid)
);

-- тестовый пользователь
insert into users
(login, password_hash)
values
('alice', encode(digest('secret', 'md5'),'hex'))
on conflict do nothing;
```

Дополнительные вопросы и задачи:

- Что можно улучшить в схеме БД?

- Доработайте механизм авторизации таким образом, что бы в каждый момент времени у пользователя была активна только одна (последняя) сессия.

- Ограничьте максимальное время пользовательской сессии до 24-х часов.

- Добавьте в БД данные об IP адресе авторизованного пользователя.

- Реализуйте методы API для получения списка всех закаченных файлов

- Реализуйте методы API для удаления файлов.

- Реализуйте работу сервера по протоколу HTTPS.