В системе будут использоваться четыре пользователя:

- web_anon - гость, который может просматривать все таблицы в публичном доступе (scheme public):
  - collection;
  - age;
  - family;
  - sex;
  - country;
  - voucher_institute;
  - region;
  - genus;
  - collector_to_collection;
  - order;
  - collector;
  - subregion;
  - kind;
  - collection;
- lab_worker - работник лаборатории, имеет те же привелегии, что и web_anon, но также может добавлять и изменять любые таблицы в публичном доступе (scheme public), а также функции добавления и изменения записей в таблице collection;
- head_lab - заведующий лабораторией, имеет те же привелегии, что и lab_worker, но также может выполнять все функции в публичном доступе (scheme public) и все процедуры в авторизации (scheme auth), что позволяет ему:
  - добавлять пользователя;
  - удалять пользователя;
  - изменять пользователя;
- database_admin - администратор базы данных обладает всеми привелегиями на обе схемы (auth и public).

```
-- Настройка web_anon - роль для гостей
create role web_anon nologin;
grant usage on schema public to web_anon;
```

```
grant select on all tables in schema public to web_anon;
grant execute on function public.login(text, text) to web_anon;

-- работник лаборатории, может выполнять процедуры, что позволяет ему изменять базу
↪  данных
create user lab_worker;
grant web_anon to lab_worker;
grant USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public to lab_worker;
grant execute on function add_collection(text, text, varchar, varchar, varchar,
↪  varchar, varchar, text, text, text,
                                    geography, text, text, text, text, date, text,
                                    ↪  text[], boolean) TO lab_worker;
grant execute on function update_collection_by_id(int, text, text, varchar, varchar,
↪  varchar, varchar, varchar, text, text, text,
                                    geography, text, text, text, text, date, text,
                                    ↪  text[], boolean) TO lab_worker;
grant insert, update on all tables in schema public to lab_worker;
revoke delete on all tables in schema public from lab_worker;
REVOKE execute on function remove_collection_by_id(int) FROM lab_worker;



-- зав. лабораторией
create user head_lab;
grant lab_worker to head_lab;
grant execute on all functions in schema public to head_lab;
grant usage on schema auth to head_lab;
grant insert, update, delete, select on all tables in schema public to head_lab;
grant insert, update, delete, select on all tables in schema auth to head_lab;

-- администратор бд - очень доверенное лицо, которое будет уверено в своих запросах
create user database_admin;
grant all privileges on schema public to database_admin;
grant all privileges on schema auth to database_admin;
```

Также дополнительно реализована роль аутентификатор(authenticator), у этой роли есть права всех пользователей, выше, благодоря этому модуль postgres реализовывает распределение ролями.

```
create role authenticator noinherit login password 'abobapass';
grant web_anon to authenticator;
grant lab_worker to authenticator;
grant head_lab to authenticator;
```

# 1 Авторизация

Для авторизации была реализована отдельная таблица users в схеме auth, код которой приведён ниже:

```
create table if not exists
    auth.users
(
    email text primary key check ( email ~* '^.+@.+\..+$' ),
    pass  text not null check (length(pass) < 512),
    role  name not null check (length(role) < 512)
);
```

Реализован триггер, который проверяет, существует ли роль, которую мы хотим записать.

```
create or replace function
    auth.check_role_exists() returns trigger as
$$
begin
    if not exists (select 1 from pg_roles as r where r.rolname = new.role) then
        raise foreign_key_violation using message =
                    'unknown database role: ' || new.role;
        -- return null;
    end if;
    return new;
end
$$ language plpgsql;

drop trigger if exists ensure_user_role_exists on auth.users;
create constraint trigger ensure_user_role_exists
    after insert or update
    on auth.users
    for each row
execute procedure auth.check_role_exists();
```

Также реализован триггер шифрования паролей, он использует расширение pgcrypto, а именно функции:

– crypt - функция, которая шифрует строку с солью, собственным алгоритмом, который работает быстрее и безопаснее, чем алгоритмы sha-1 и md-5, но при этом основывается на них.

  Также данная функция поддерживает проверку пароля не расскрывая соль, которая использовалася при генерации, что обеспечивает дополнительную безопасность;

– gen_salt - функция, которая генерирует соль и указывает основной алгоритм шифрования данных.

Источник

```
create or replace function
    auth.encrypt_pass() returns trigger as
$$
begin
    if tg_op = 'INSERT' or new.pass <> old.pass then
        new.pass = crypt(new.pass, gen_salt('bf'));
    end if;
    return new;
end
$$ language plpgsql;

drop trigger if exists encrypt_pass on auth.users;
create trigger encrypt_pass
    before insert or update
    on auth.users
    for each row
execute procedure auth.encrypt_pass();
```

| | email | pass | role |
|---|---|---|---|
| 1 | pank@pank.su | $2a$06$p1ZRJ1tWTNMnaifHDIDahO4XA6eMldwylFRcFgJ16TX5xeJc0h… | head_lab |

Рисунок 1 – Демонстрация хеширования

Реализована функция получение роли по почте и паролю:

```
create or replace function
    auth.user_role(email text, pass text) returns name
    language plpgsql
as
$$
begin
    return (select role
            from auth.users
            where users.email = user_role.email
              and users.pass = crypt(user_role.pass, users.pass));
end;
$$;
```

И безопасная функция, которой будет пользоваться postgrest для получения jwt-токена:

```
CREATE TYPE auth.jwt_token AS
(
    token text
);

create extension if not exists pgjwt;


ALTER DATABASE lab_base SET "app.jwt_secret" TO
↪  'Q5He86xPvYscMiZxQw29gy8YkbD7a4aMDH1hQFP';


-- функция авторизации, для анонимного пользователя нужно только выполненные функций
create or replace function
    public.login(email text, pass text) returns auth.jwt_token as
$$
declare
    _role  name;
    result auth.jwt_token;
begin
    -- check email and password
    select auth.user_role(email, pass) into _role;
    if _role is null then
        raise invalid_password using message = 'invalid user or password';
    end if;
    -- НЕ КОМУ НЕ СООБЩАТЬ КОД, НЕ ХРАНИТЬ ЕГО В ОТКРЫТЫХ ПЕРЕМЕННЫХ
    select sign(
                   row_to_json(r), current_setting('app.jwt_secret')
               ) as token
    from (select _role                                          as role,
                 $1                                             as email,
                 extract(epoch from now())::integer + 60 * 60 as exp) r
    into result;
    return result;
end;
$$ language plpgsql security definer;
```

Настройки postgrest:

```
db-uri = "postgres://authenticator:abobapass@localhost:5432/lab_base"
db-schemas = "public"
db-anon-role = "web_anon"

jwt-secret = "Q5He86xPvYscMiZxQw29gy8YkbD7a4aMDH1hQFP"
```

# 2 Демонстрация работы

## 2.1 web_anon

Получение записи по id=1 в коллекции
HTTP Запрос:

```
GET http://localhost:3000/collection?id=eq.1
```

Аналогичный SQL запрос

```
SELECT * FROM collection WHERE id = 1;
```

Результат выполнения:


```
[
  {
    "id": 1,
    "CatalogueNumber": "ZIN-TER-M-55",
    "collect_id": "1",
    "kind_id": 1,
    "subregion_id": 1,
    "gen_bank_id": null,
    "point": "0101000020E6100000CACDE459D130554008D8BBFD41264940",
    "vouch_inst_id": 1,
    "vouch_id": "91130",
    "rna": false,
    "sex_id": 2,
    "age_id": 3,
    "day": 12,
    "month": 6,
    "year": 2005,
    "comment": "По сиквенсам co1 и цитохрома - красно-серая, первоначально определена
    ↪ как красная JF713496",
    "geo_comment": "Алтай, Республика, Усть-Коксинский р-он, Банное, село, р. Колчулу"
  }
]
```


Попытка добавить в age запись:
HTTP запрос:

```
POST http://localhost:3000/age
Content-Type: application/json
{"name": "test age"}
```

Ответ:

```
{
  "code": "42501",
  "details": null,
  "hint": null,
  "message": "нет доступа к таблице age"
}
```

Демоннстрация, что age - доступна:
HTTP запрос:

```
GET http://localhost:3000/age
```

Ответ:

```
[
  {
    "id": 0,
    "name": "Unknown"
  },
  {
    "id": 2,
    "name": "subadult"
  },
  {
    "id": 3,
    "name": "adult"
  },
  {
    "id": 1,
    "name": "juvenile"
  },
  {
    "id": 4,
    "name": "subadult or adult"
  },
  {
    "id": 19,
    "name": "test"
  }
]
```

## 2.2 lab_worker

Добавление тестового пользователя с ролью lab_worker в таблицу users:

```
INSERT INTO auth.users(email, pass, role) VALUES ('test@test.com', 'test',
↪  'lab_worker');
```

Демонстрация авторизации:
Хороший пароль
HTTP запрс:

```
POST http://localhost:3000/rpc/login
Content-Type: application/json

{"email": "test@test.com", "pass":"test"}
```

Ответ:

```
{
  "token":
  ↪  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0L
}
```

Плохой пароль
HTTP запрос:

```
POST http://localhost:3000/rpc/login
Content-Type: application/json

{"email": "test@test.com", "pass":"я плохой пароль"}
```

Ответ

```
{
  "code": "28P01",
  "details": null,
  "hint": null,
  "message": "invalid user or password"
}
```

Добавление записи за lab_worker:
HTTP запрос:

```
POST http://localhost:3000/rpc/add_collection
Authorization: Bearer
↪  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0LmNv
Content-Type: application/json

{ "catalog_number": "test", "collect_id": "test", "order" : "test", "family" : "test",
                    "genus" : "test", "kind" : "test", "age" : "test", "sex" : "test",
                    ↪  "vauch_inst" : "test",
                    "vauch_id" : "test", "point" : "POINT(0 0)", "country" : "test",
                    ↪  "region" : "test", "subregion" : "test",
                    "geocomment": "test", "date_collect" : "2004-4-4", "comment" : "Это
                    ↪  тестовая запись",
                    "collectors" : "{{\"Панков\", \"\", \"\" }, {\"Турсунова\", \"\",
                    ↪  \"\"}}"}
```

Ответ пустой.
Проверка, что запись добавилась:
HTTP запрос:

```
GET http://localhost:3000/collection?id=eq.6081
```

Ответ:

```
[
  {
    "id": 6081,
    "CatalogueNumber": "test",
    "collect_id": "test",
    "kind_id": 270,
    "subregion_id": 154,
    "gen_bank_id": null,
    "point": "0101000020E6100000000000000000000000000000000000",
    "vouch_inst_id": 46,
    "vouch_id": "test",
    "rna": false,
    "sex_id": 18,
```

```
    "age_id": 19,
    "day": 4,
    "month": 4,
    "year": 2004,
    "comment": "Это тестовая запись",
    "geo_comment": "test"
  }
]
```

Попытка удаления записи:
HTTP запрос:

```
POST http://localhost:3000/rpc/remove_collection_by_id
Authorization: Bearer
↪  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoibGFiX3dvcmtlciIsImVtYWlsIjoidGVzdEB0ZXN0LmNv
Content-Type: application/json

{"col_id": 6081}
```

Ответ:

```
{
  "code": "42501",
  "details": null,
  "hint": null,
  "message": "нет доступа к таблице collection"
}
```

## 2.3 head_lab

Добавление тестового пользователя с ролью lab_worker в таблицу users:

```
INSERT INTO auth.users(email, pass, role) VALUES ('test2@test.com', 'test',
↪  'head_lab');
```

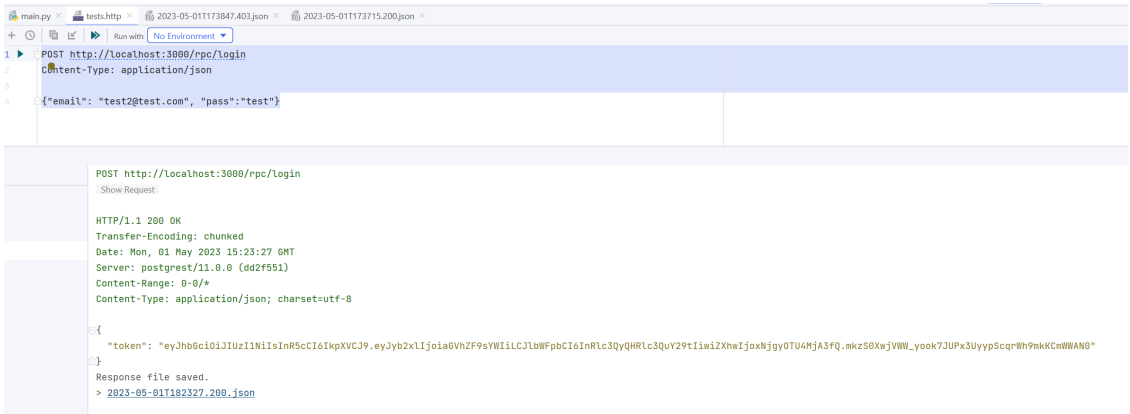


Рисунок 2 – Авторизация

Попытка удаления
HTTP запрос:

```
POST http://localhost:3000/rpc/remove_collection_by_id
Authorization: Bearer
↪  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJyb2xlIjoiaGVhZF9sYWIiLCJlbWFpbCI6InRlc3QyQHRlc3QuY29t
```

```
Content-Type: application/json

{"col_id": 6081}
```

Ответ

```
<Response body is empty>
```

Демонстрация, что запись удалена:
HTTP запрос:

```
GET http://localhost:3000/collection?id=eq.6081
```

Ответ:

```
[]
```

Вывод: запись удалена