

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

# **Лабораторна робота №1**

***з дисципліни «Бази даних»***

**«Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL»**

Виконав студент групи: КВ-33

ПІБ: Ляшенко Д. Д.

Перевірив: Павловський В. І.

**Київ 2025**

**Постановка задачі**

В рамках даної лабораторної роботи мають бути вирішені наступні задачі:

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

***Завдання №1:***

**Опис предметної галузі**

Тема бази даних: Онлайн-платформа для здачі та оренди нерухомості. На заданій платформі людина (користувач) може одночасно виступати як орендарем, так і тим, хто здає нерухомість в оренду. Користувач може як бронювати, так і здавати в оренду нерухомість. Нерухомість може мати різні відгуки від користувачів. У даної моделі платформи передбачені умови, за які нерухомість може бути орендованою.

**Розробка концептуальної моделі**

**При створенні даної бази даних я виділив такі сутності:**

1. **Користувач (User)** – представляє клієнтів платформи:

* Атрибути: ім'я, прізвище, електронна пошта, дата реєстрації.

1. **Нерухомість (Realty)** – представляє об’єкт нерухомості (квартира, будинок, тощо):

* Атрибути: тип нерухомості (квартира, будинок, тощо), назва міста де знаходиться нерухомість, вулиця місцезнаходження нерухомості, статус нерухомості (здається в оренду, не здається в оренду, орендується), мінімальний строк оренди, депозит(0, сума депозиту), дозволені умови (примітка від власника нерухомості), ціна, термін внесення цінової політики (місяць, доба).

1. **Бронювання (Booking)** – представляє можливість орендування нерухомості:

* Атрибути: дата початку, дата завершення, статус (очікує підтвердження, підтверджено, відхилено, завершено), сума угоди.

1. **Відгуки (Review)** – представляє об’єкт оцінки нерухомості:

* Атрибути: рейтинг (1-5).

1. **Власник нерухомості (Property owner)** – представляє собою користувача, що здає нерухомість в оренду:

* Атрибути: ім’я, прізвище, електронна пошта, дата реєстрації.

**Зв’язки:**

Зв’язок «Власник» - «Нерухомість»:

- Тип зв'язку: 1 до N (один власник може мати безліч нерухомостей; одна нерухомість відноситься до одного власника).

Зв’язок «Нерухомість» - «Відгук»:

- Тип зв'язку: 1 до N (одна нерухомість може мати безліч відгуків; один відгук відноситься до однієї нерухомості).

Зв’язок «Користувач» - «Відгук»:

- Тип зв'язку: 1 до N (один користувач може залишати безліч відгуків; один відгук належить одному користувачу).

Зв’язок «Власник» - «Нерухомість» - «Користувач»:

- Даний зв’язок є сутністю з ім’ям Бронювання та відповідними йому атрибутами. Даний зв’язок є неділимим. Сенс цього зв’язку існує лише у повному наборі сутностей, які в ньому беруть участь. Розглянемо 3 зв’язки між 2 сутностями, які ми можемо отримати в рамках даного зв’язку:

Зв’язок «Власник» - «Бронювання»:

- Тип зв’язку: 1 до N (один власник може мати багато бронювань; одне бронювання належить конкретному власнику).

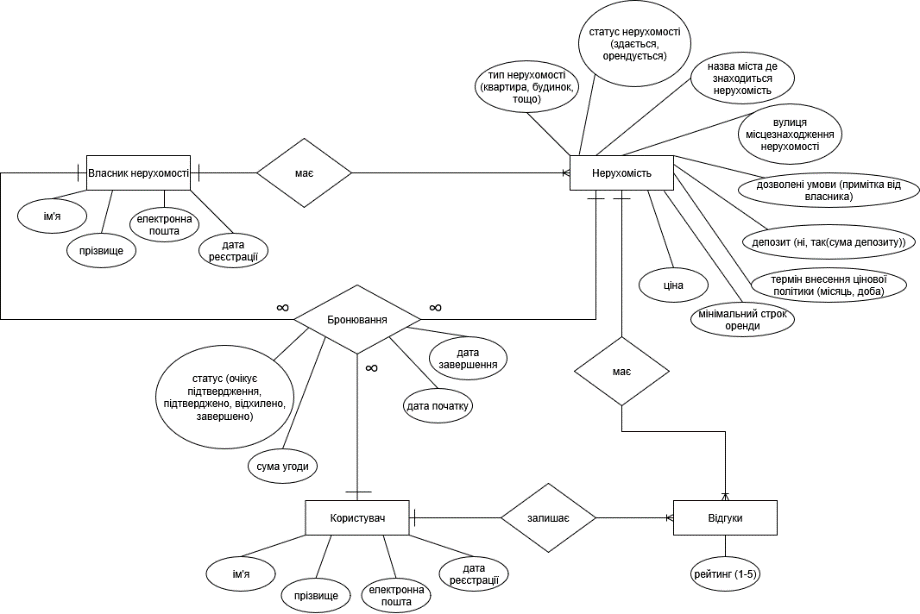
Зв’язок «Нерухомість» - «Бронювання»:

- Тип зв’язку: 1 до N (одна нерухомість може бути заброньованою багато разів; одне бронювання відноситься до однієї конкретної нерухомості).

Зв’язок «Користувач» - «Бронювання»:

- Тип зв’язку: 1 до N (один користувач може робити багато бронювань; одне бронювання відноситься до одного користувача).

Графічне подання концептуальної моделі «Сутність-зв’язок» (ER-діаграма, побудована за нотацією “Пташина лапка”) зображено на рисунку:



***Перетворення концептуальної моделі у логічну модель та схему бази даних***

Сутність "Користувач" перетворено в таблицю User з первинним ключем user\_id та атрибутами first\_name, last\_name, data\_registration, email.

Сутність "Нерухомість" перетворено в таблицю Realty з первинним ключем realty\_ id та атрибутами type\_realty, status\_realty, city\_name, street\_name. У таблиці є зовнішній ключ user\_id, який пов’язаний з таблицею Property owner.

Сутність "Умови оренди" перетворено в таблицю Rental Terms з первинним ключем rental\_terms\_id та атрибутами minimum\_rental\_period, deposit, permitted\_conditions, price, payment\_term. У таблиці є зовнішні ключі user\_id та realty\_id, які пов’язані із таблицями Property owner та Realty відповідно.

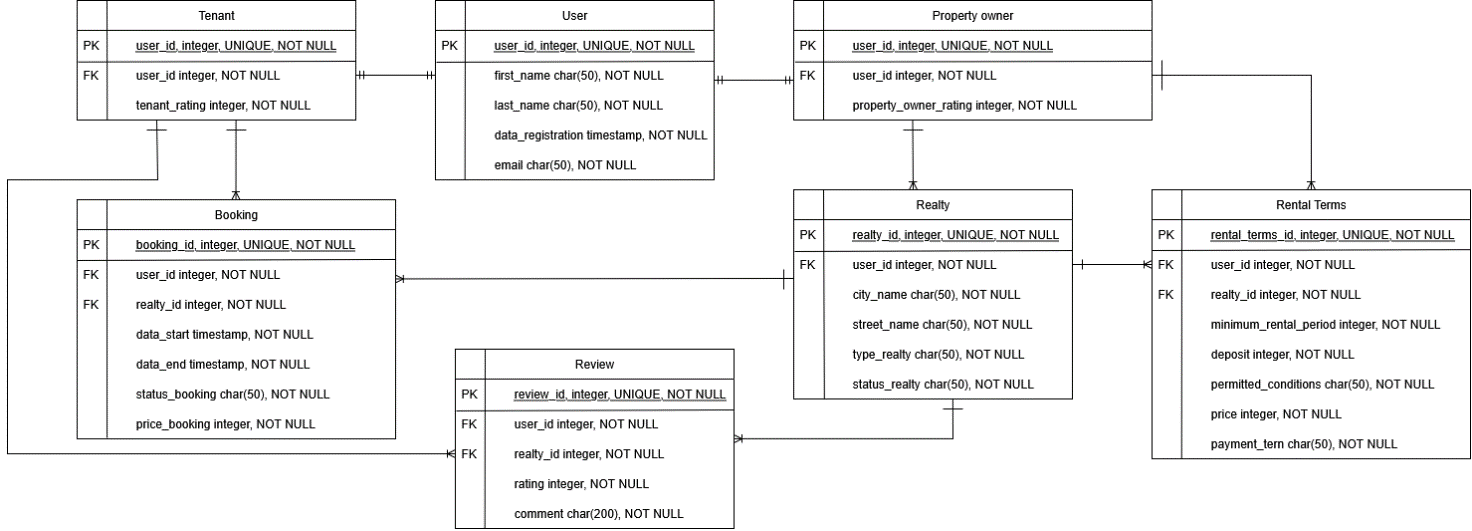
Сутність "Бронювання" перетворено в таблицю Booking з первинним ключем booking\_id та атрибутами data\_start, data\_end, status\_booking, price\_booking. У таблиці є зовнішні ключі user\_id та realty\_id, які пов’язані з таблицями Tenant та Realty відповідно.

Сутність “Відгуки” перетворено в таблицю Review з первинним ключем review\_id та атрибутами rating, comment. У таблиці є зовнішні ключі user\_id та realty\_id, які пов’язані з таблицями Tenant та Realty відповідно.

Сутність “Орендар” перетворено в таблицю Tenant з первинним ключем user\_id та атрибутами tenant\_rating. У таблиці є зовнішній ключ user\_id, який пов’язаний з таблицею User.

Сутність “Власник нерухомості” перетворено в таблицю Property owner з первинним ключем user\_id та атрибутами property\_owner\_rating. У таблиці є зовнішній ключ user\_id, який пов’язаний з таблицею User.

Графічне подання логічної моделі «Сутність-зв’язок» зображено на рисунку:



*Схема бази даних*

Таблиця ілюструє детальний перехід від однієї моделі до іншої.

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| *User* – містить дані про користувача | *user\_id –* унікальний ідентифікатор користувача | integer (числовий) |
| *first\_name –* ім’я користувача | character varying (рядок) |
| *last\_name –* прізвище користувача | character varying (рядок) |
| *data\_registration –* дата реєстрації користувача | timestamp (дата та час) |
| *email –* електронна скринька користувача | character varying (рядок) |
| *Realty –* містить дані про нерухомість | *realty\_id* – унікальний ідентифікатор нерухомості | integer (числовий) |
| *user\_id –* ідентифікатор власника нерухомості | integer (числовий) |
| *type\_realty –* тип нерухомості | character varying (рядок) |
| *status\_realty –* нерухомість здається чи орендується | character varying (рядок) |
| *city\_name –* назва населеного пункту де знаходиться нерухомість | character varying (рядок) |
| *street\_name –* назва вулиці де знаходиться нерухомість | character varying (рядок) |
| *Rental Terms –* містить дані умов, за які нерухомість може бути орендованою | *rental\_terms\_id -* унікальний ідентифікатор умови | integer (числовий) |
| *minimum\_rental\_period –* мінімальний строк оренди | character varying (рядок) |
| *deposit –* депозит | integer (числовий) |
| *permitted\_conditions –* коментар від власника нерухомості | character varying (рядок) |
| *price –* ціна | integer (числовий) |
| *payment\_term –* строк до котрого плата має бути внесена | character varying (рядок) |
| *user\_id –* ідентифікатор власника нерухомості | integer (числовий) |
| *realty\_id –* ідентифікатор нерухомості | integer (числовий) |
| *Booking –* містить дані про бронювання орендарями нерухомості | *booking\_id –* унікальний ідентифікатор бронювання | integer (числовий) |
| *user\_id –* ідентифікатор орендаря | integer (числовий) |
| *realty\_id –* ідентифікатор нерухомості | integer (числовий) |
| *data\_start –* дата початку бронювання | timestamp (дата та час) |
| *data\_end –* дата завершення бронювання | timestamp (дата та час) |
| *status\_booking –* статус бронювання підтверджено, завершено, відхилено чи очікує підтвердження | character varying (рядок) |
| *price\_booking –* загальна сума угоди | integer (числовий) |
| *Review –* містить дані щодо оцінки нерухомості | *review\_id –* унікальний ідентифікатор оцінки | integer (числовий) |
| *user\_id –* ідентифікатор орендаря | integer (числовий) |
| *realty\_id –* ідентифікатор нерухомості | integer (числовий) |
| *rating –* оцінка нерухомості | integer (числовий) |
| *comment –* коментар від орендаря | character varying (рядок) |
| *Tenant –* містить дані орендаря | *user\_id –* унікальний ідентифікатор орендаря та ідентифікатор користувача | integer (числовий) |
| *tenant\_rating –* рейтинг орендаря | integer (числовий) |
| *Property owner –* містить дані власника нерухомості | *user\_id –* унікальний ідентифікатор власника нерухомості та ідентифікатор користувача | integer (числовий) |
| *property\_owner\_rating –* рейтинг власника нерухомості | integer (числовий) |

**Функціональні залежності для кожної таблиці**

1. User:

user\_id -> first\_name, last\_name, data\_registration, email

2. Realty:

realty\_id -> user\_id, type\_realty, status\_realty, city\_name, street\_name

user\_id -> type\_realty, status\_realty, city\_name, street\_name

3. Rental Terms:

rental\_terms\_id -> user\_id, realty\_id, minimum\_rental\_period, deposit, permitted\_conditions, price, payment\_term

user\_id -> realty\_id, minimum\_rental\_period, deposit, permitted\_conditions, price, payment\_term

realty\_id -> user\_id, minimum\_rental\_period, deposit, permitted\_conditions, price, payment\_term

4. Booking:

booking\_id -> user\_id, realty\_id, data\_start, data\_end, status\_booking, price\_booking

user\_id -> realty\_id, data\_start, data\_end, status\_booking, price\_booking

realty\_id -> user\_id, data\_start, data\_end, status\_booking, price\_booking

5. Review:

review\_id -> user\_id, realty\_id, rating, comment

user\_id -> realty\_id, rating, comment

realty\_id -> user\_id, rating, comment

6. Tenant:

user\_id -> tenant\_rating

7. Property owner

user\_id -> property\_owner\_rating

Дані функціональні залежності показують, які атрибути в таблиці можуть бути однозначно визначені на основі інших атрибутів у моделі бази даних. Це відіграє ключову роль у процесі нормалізації та правильній організації даних, оскільки коректне встановлення таких залежностей допомагає уникнути дублювання інформації та зберегти її цілісність.

Розглядаючи саму суть транзитивних функціональних залежностей можна стверджувати, що вони з’являються за умови, коли значення одного атрибута визначається через проміжний атрибут. Інакше кажучи, якщо атрибут А визначає атрибут В, а В у свою чергу визначає атрибут С, то можна зробити висновок, що А транзитивно визначає С. У даній представленій моделі таких транзитивних залежностей не виявлено, що в свою чергу свідчить про відсутність надлишкових даних і забезпечує оптимальне функціонування бази.

**Відповідність схеми нормальним формам**

1. Щоб задовільнити умови 1НФ кожен атрибут в таблиці має бути атомарним, тобто:

* Кожна клітинка містить єдине значення;
* Кожен запис є унікальним.

У моїй створеній моделі бази даних усі атрибути мають загальну неподільну структуру, в одному полі зберігається виключно одне єдине значення. Унікальність кожного рядка зберігається завдяки використаним первинним ключам. Отже, виходячи з вищеописаного, схема ***відповідає вимогам 1НФ.***

1. Щоб схема відповідала 2НФ повинні виконуватись умови:

* Схема перебуває в 1НФ;
* Кожний неключовий атрибут функціонально залежить від цілого ключа.

Дана створена мною база даних знаходиться у 1НФ. Усі атрибути даної бази, що не є ключовими, визначаються первинним ключем у кожній окремій таблиці, а отже, часткові залежності відсутні. Для прикладу можна розглянути таблицю “User”, де усі поля, що не являються ключовими залежать від первинного ключа. Отже, виходячи з вищеописаного, схема ***відповідає 2НФ.***

1. Щоб схема відповідала 3НФ повинні виконуватись умови:

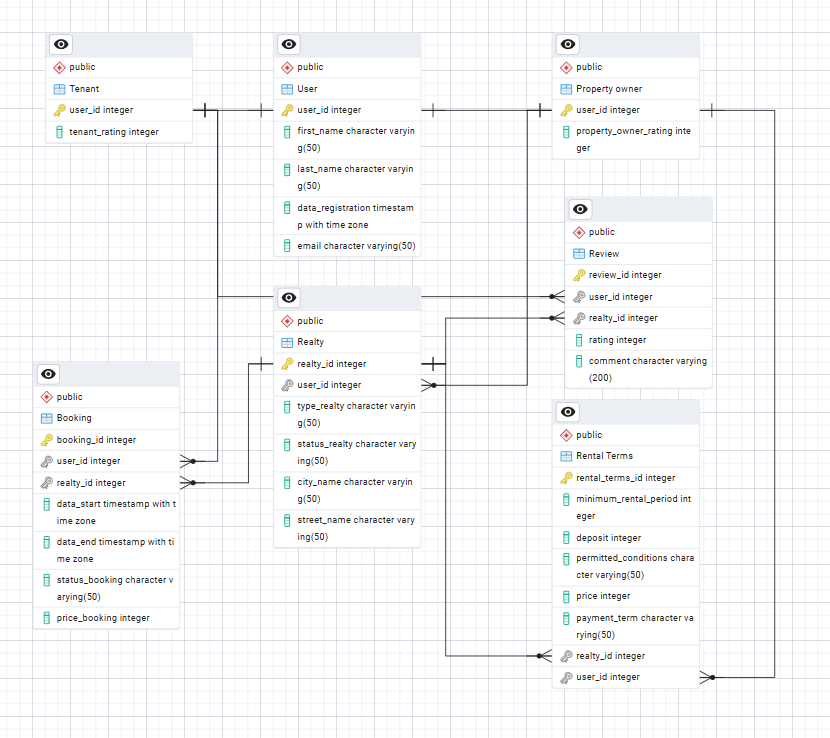
* Схема перебуває в 2НФ;
* Кожен не первинний атрибут має бути не транзитивно залежним від кожного ключа.

Виходячи з того факту, що схема перебуває у 2НФ та транзитивні залежності між неключовими атрибутами у ній відсутні, а також, усі неключові поля залежать від первинного ключа, можна до прикладу розглянути таблицю “Review”, де жоден неключовий атрибут не визначається через інший атрибут опосередковано. Отже, виходячи з вищеописаного, схема ***відповідає 3НФ***.

Висновок:

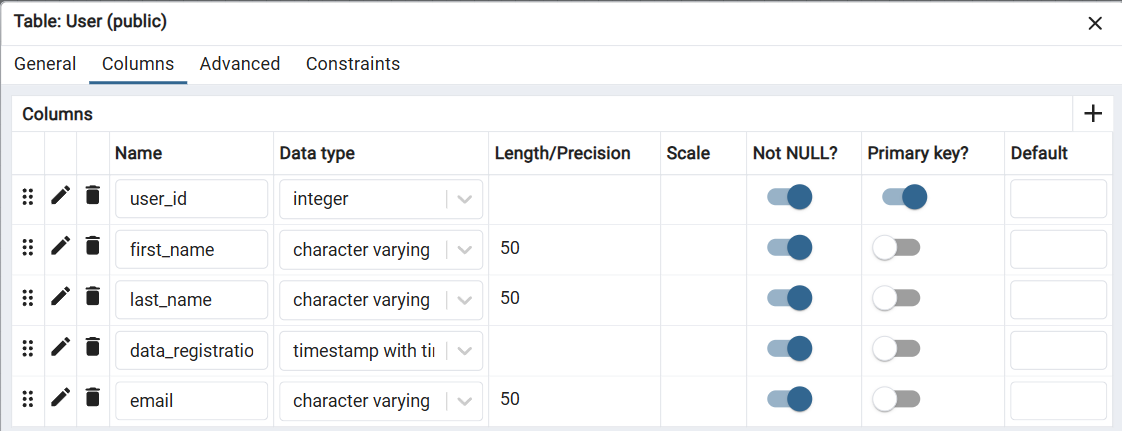
Схема бази даних відповідає нормальним формам НФ1, НФ2 та НФ3.

Схема бази даних у pgAdmin 4 зображено на рисунку:



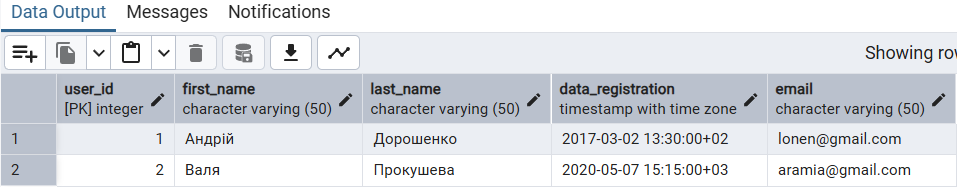
**Таблиці бази даних у pgAdmin4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**User\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

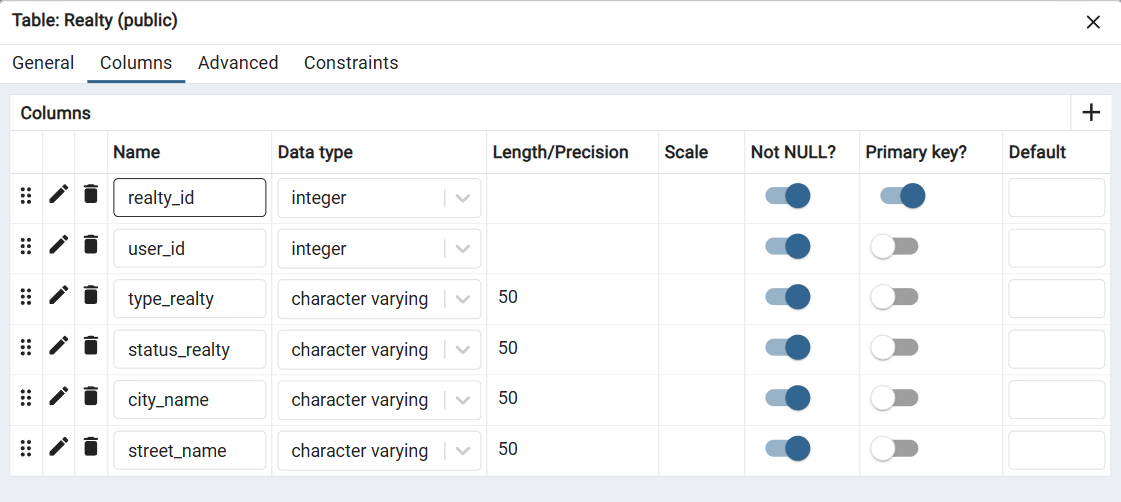


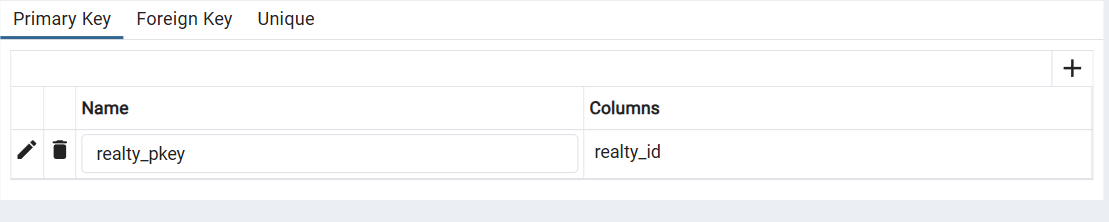


У цієї таблиці немає зовнішніх ключів (FK)

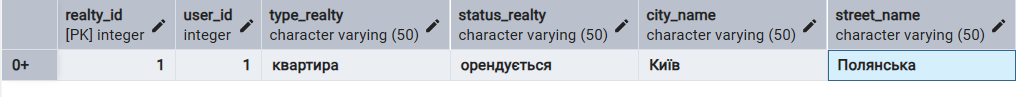


\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Realty\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

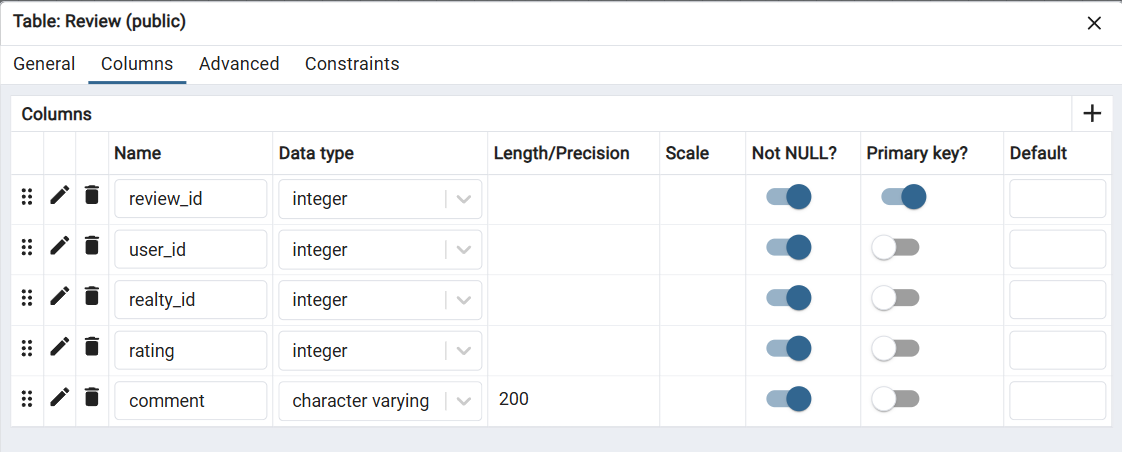


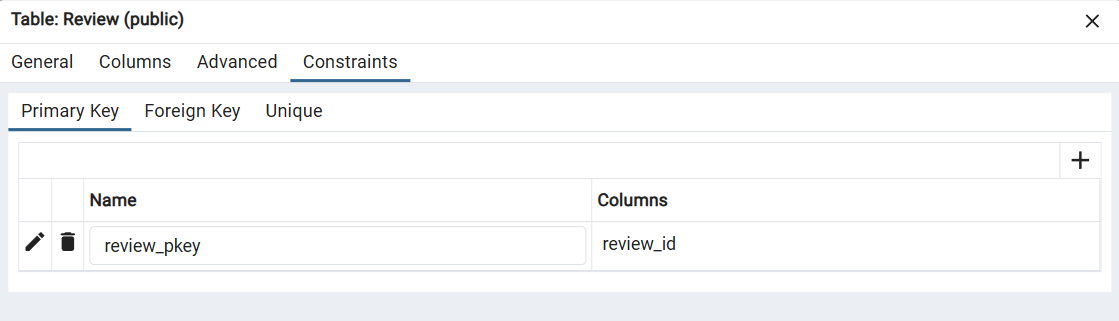






\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Review\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

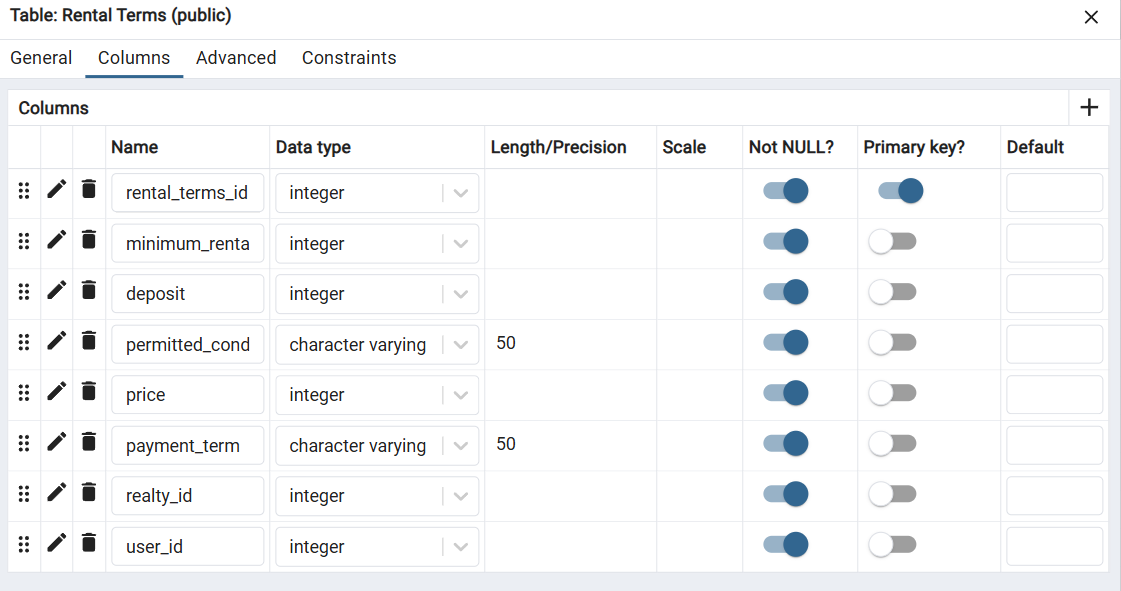
****

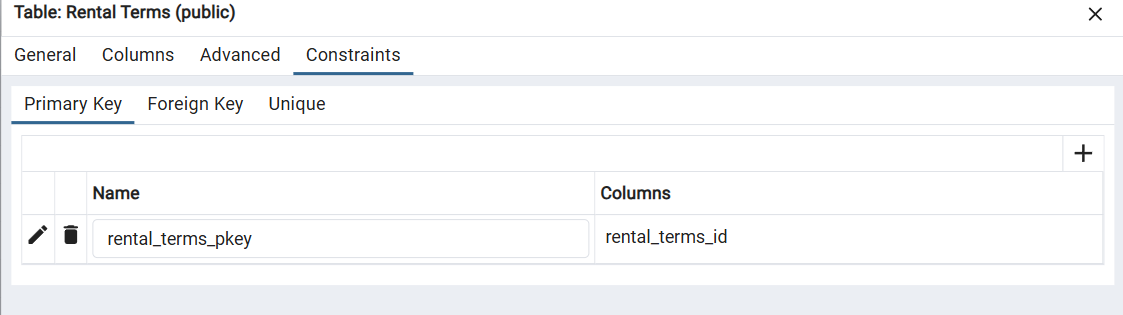
****

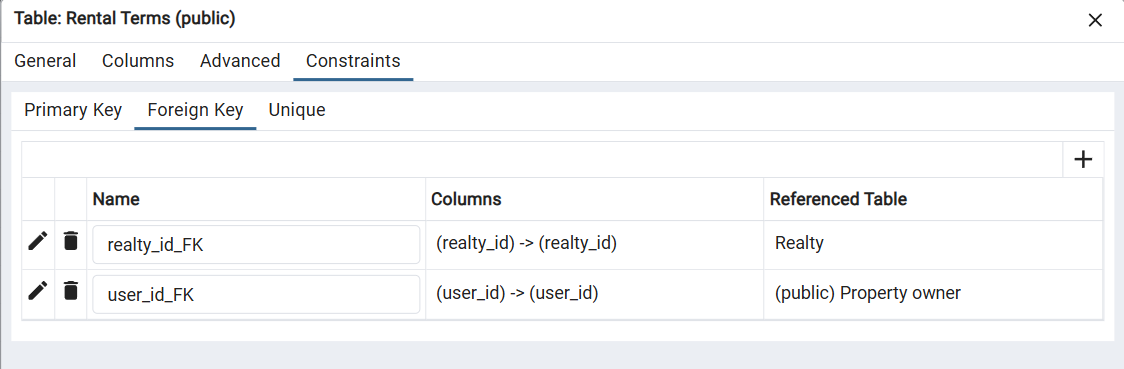
****

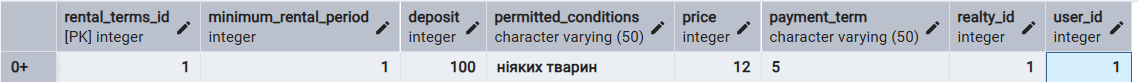
****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Rental\_Terms\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

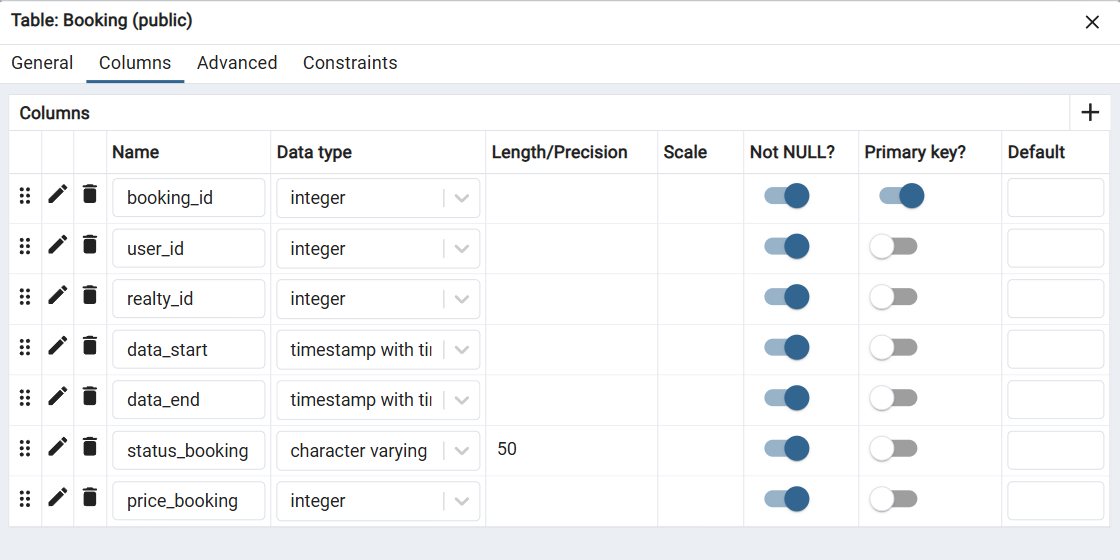
****

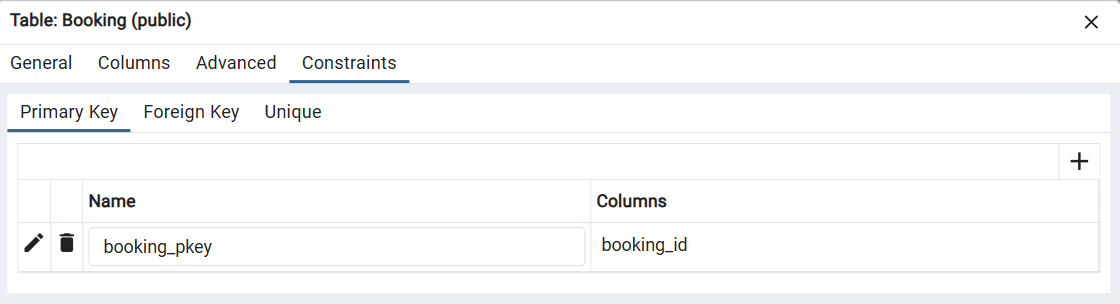
****

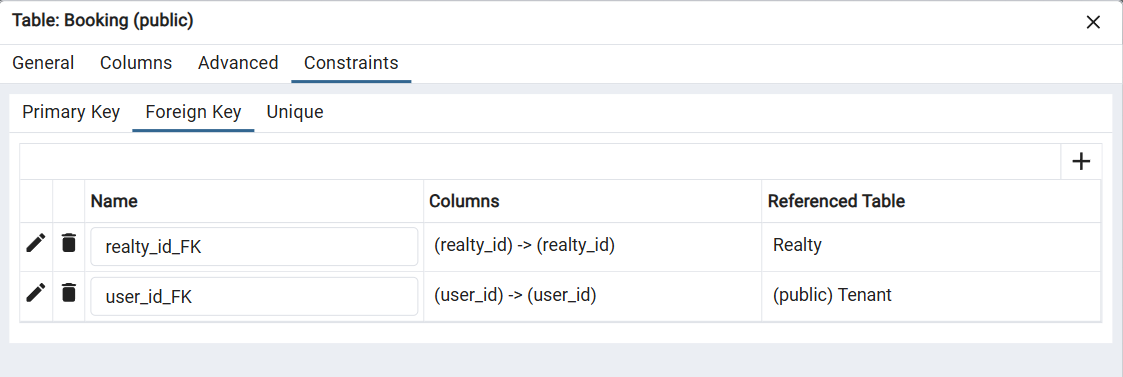


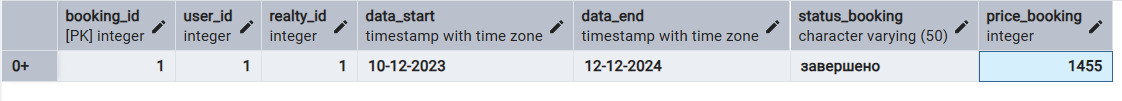
****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Booking\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

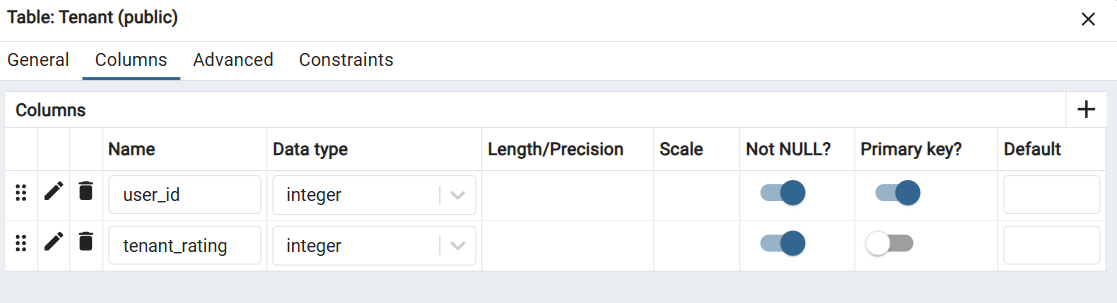
****

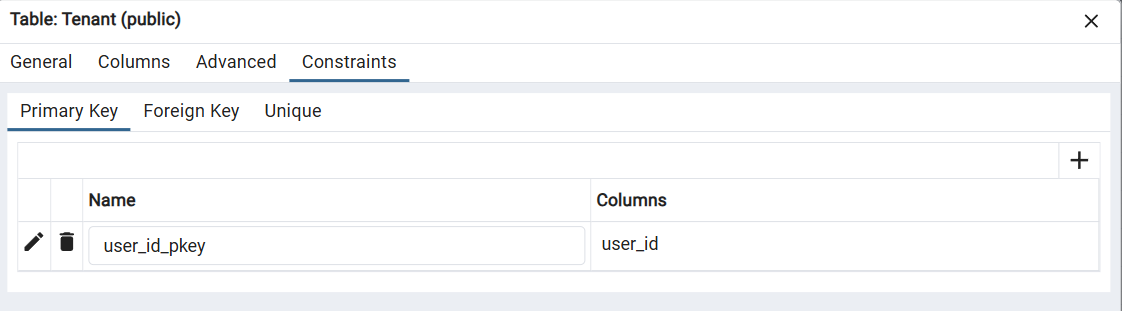
****

****

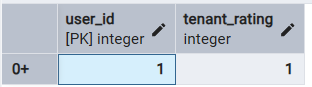
****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Tenant\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

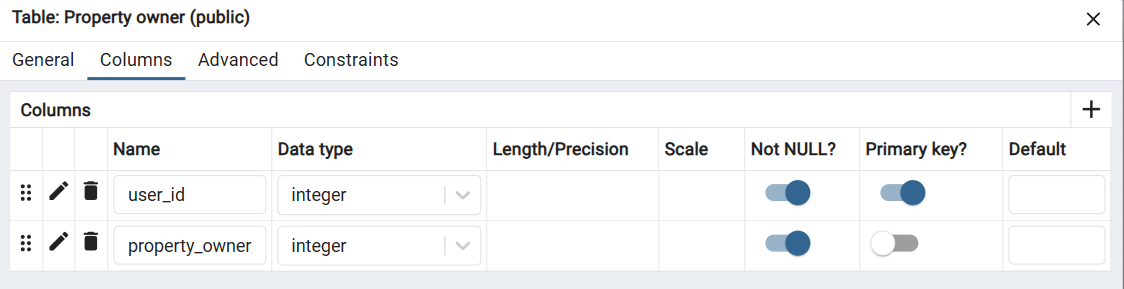
****

****

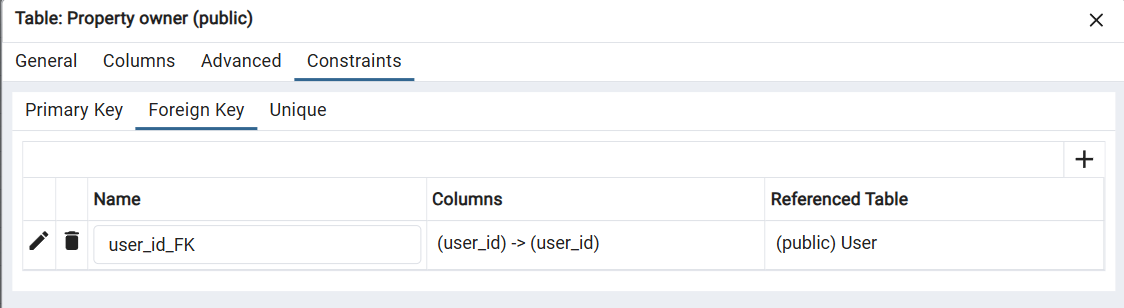
****

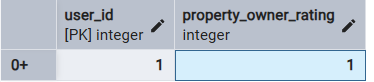
****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Property\_owner\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

****

****

****

**Таблиці в коді SQL**

-- This script was generated by the ERD tool in pgAdmin 4.

-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.

BEGIN;

CREATE TABLE IF NOT EXISTS public."User"

(

user\_id integer NOT NULL,

first\_name character varying(50) NOT NULL,

last\_name character varying(50) NOT NULL,

data\_registration timestamp with time zone NOT NULL,

email character varying(50) NOT NULL,

CONSTRAINT user\_pkey PRIMARY KEY (user\_id)

);

CREATE TABLE IF NOT EXISTS public."Realty"

(

realty\_id integer NOT NULL,

user\_id integer NOT NULL,

type\_realty character varying(50) NOT NULL,

status\_realty character varying(50) NOT NULL,

city\_name character varying(50) NOT NULL,

street\_name character varying(50) NOT NULL,

CONSTRAINT realty\_pkey PRIMARY KEY (realty\_id)

);

CREATE TABLE IF NOT EXISTS public."Review"

(

review\_id integer NOT NULL,

user\_id integer NOT NULL,

realty\_id integer NOT NULL,

rating integer NOT NULL,

comment character varying(200) NOT NULL,

CONSTRAINT review\_pkey PRIMARY KEY (review\_id)

);

CREATE TABLE IF NOT EXISTS public."Booking"

(

booking\_id integer NOT NULL,

user\_id integer NOT NULL,

realty\_id integer NOT NULL,

data\_start timestamp with time zone NOT NULL,

data\_end timestamp with time zone NOT NULL,

status\_booking character varying(50) NOT NULL,

price\_booking integer NOT NULL,

CONSTRAINT booking\_pkey PRIMARY KEY (booking\_id)

);

CREATE TABLE IF NOT EXISTS public."Rental Terms"

(

rental\_terms\_id integer NOT NULL,

minimum\_rental\_period integer NOT NULL,

deposit integer NOT NULL,

permitted\_conditions character varying(50) NOT NULL,

price integer NOT NULL,

payment\_term character varying(50) NOT NULL,

realty\_id integer NOT NULL,

user\_id integer NOT NULL,

CONSTRAINT rental\_terms\_pkey PRIMARY KEY (rental\_terms\_id)

);

CREATE TABLE IF NOT EXISTS public."Tenant"

(

user\_id integer NOT NULL,

tenant\_rating integer NOT NULL,

CONSTRAINT user\_id\_pkey PRIMARY KEY (user\_id)

);

CREATE TABLE IF NOT EXISTS public."Property owner"

(

user\_id integer NOT NULL,

property\_owner\_rating integer NOT NULL,

CONSTRAINT user\_id\_pkey PRIMARY KEY (user\_id)

);

ALTER TABLE IF EXISTS public."Realty"

ADD CONSTRAINT "user\_id\_FK" FOREIGN KEY (user\_id)

REFERENCES public."Property owner" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Review"

ADD CONSTRAINT "realty\_id\_FK" FOREIGN KEY (realty\_id)

REFERENCES public."Realty" (realty\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Review"

ADD CONSTRAINT "user\_id\_FK" FOREIGN KEY (user\_id)

REFERENCES public."Tenant" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Booking"

ADD CONSTRAINT "realty\_id\_FK" FOREIGN KEY (realty\_id)

REFERENCES public."Realty" (realty\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Booking"

ADD CONSTRAINT "user\_id\_FK" FOREIGN KEY (user\_id)

REFERENCES public."Tenant" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Rental Terms"

ADD CONSTRAINT "realty\_id\_FK" FOREIGN KEY (realty\_id)

REFERENCES public."Realty" (realty\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Rental Terms"

ADD CONSTRAINT "user\_id\_FK" FOREIGN KEY (user\_id)

REFERENCES public."Property owner" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Tenant"

ADD CONSTRAINT user\_id FOREIGN KEY (user\_id)

REFERENCES public."User" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Property owner"

ADD CONSTRAINT "user\_id\_FK" FOREIGN KEY (user\_id)

REFERENCES public."User" (user\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

END;

Посилання на GitHub:

<https://github.com/liashenkodaniil/->

Контакти в телеграм:

Username: @daniczek

Nickname: Danchik

Електронна пошта:

kyvalich@gmail.com