# *DJANGO*

**ORM I**

# *PLAN NA DZISIAJ*

- Przechowywanie danych
- ORM  - wstęp
- ORM dla Pythona  - przegląd
- Django ORM
  - Instrukcje DDL
  - Instrukcje DML i DQL
- Panel administratora
- Aplikacja CRUD

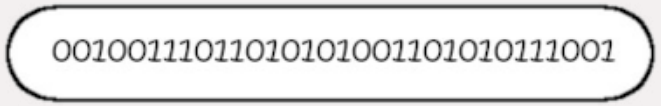
# *PRZECHOWYWANIE DANYCH*

Dane możemy przechowywać w:

1. Pamięci ulotnej komputera (klasyczna pamięć RAM)
2. Pamięci trwałej komputera
    A. W pliku (tekstowym, binarnym, …)
    B. W bazie danych

# ORM

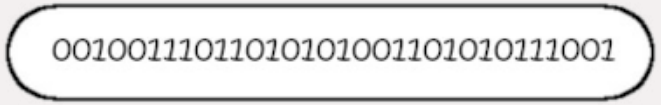
## WSTEP


00100111011010101001101010111001

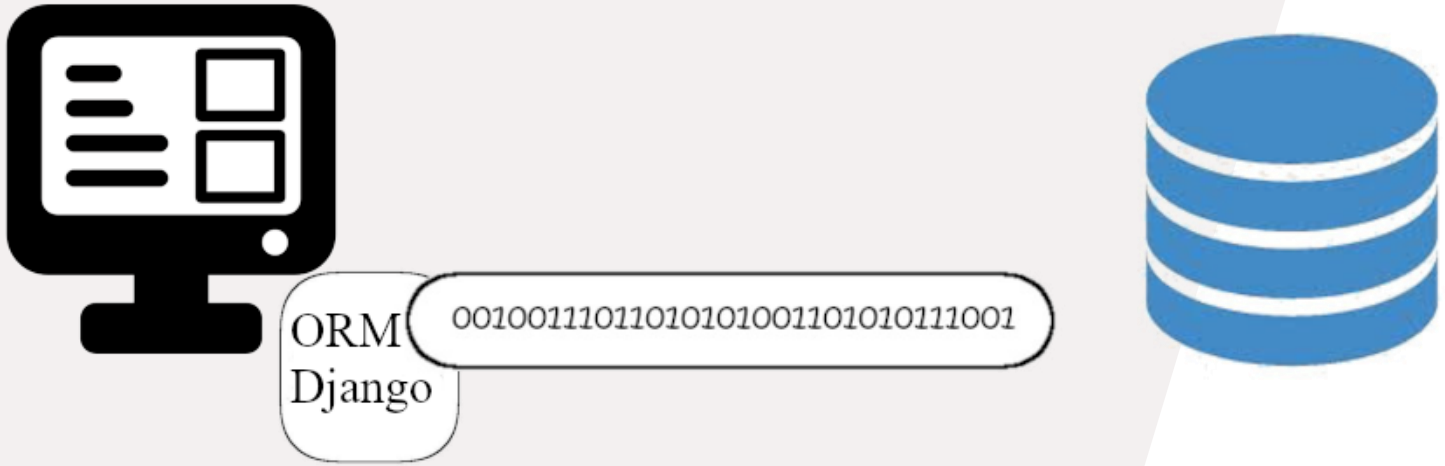

00100111011010101001101010111001

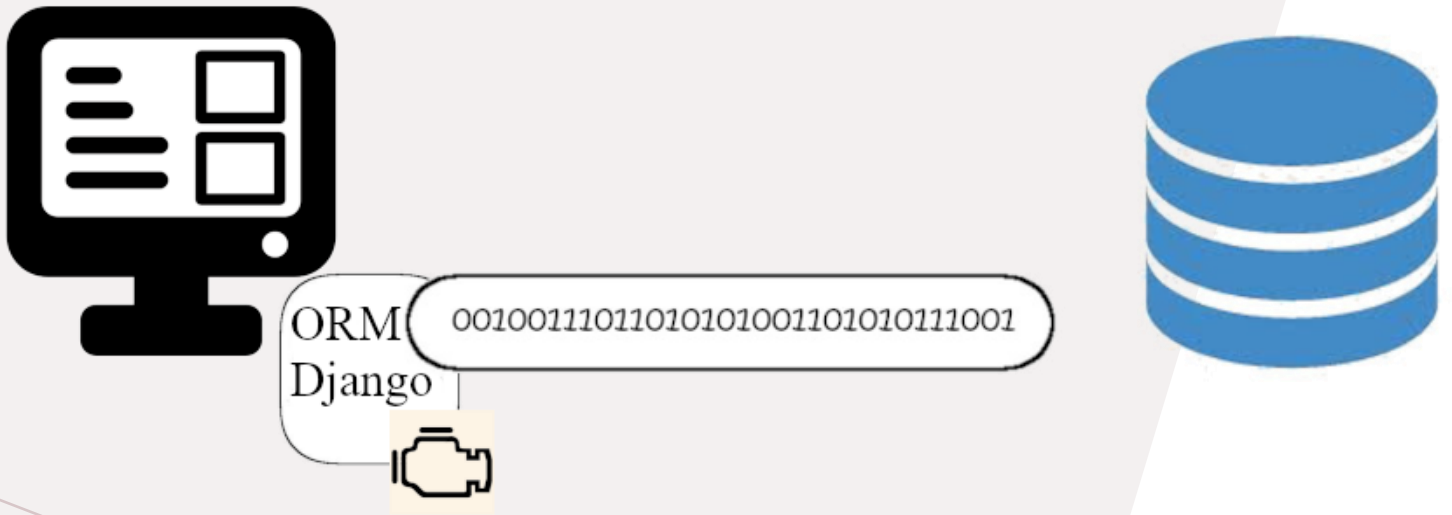
```
CREATE TABLE user
(
  user_id INT NOT NULL PRIMARY KEY,
);
```

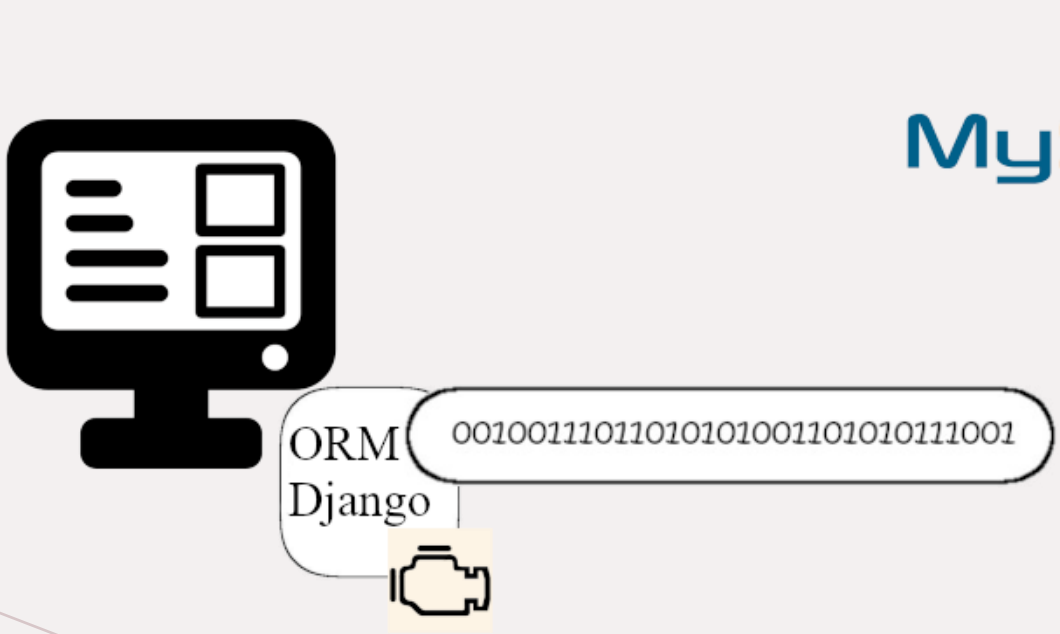
ORACLE




```
CREATE TABLE user
(
  user_id numeric(10) not null,
  CONSTRAINT user_pk PRIMARY KEY (user_id)
);
```

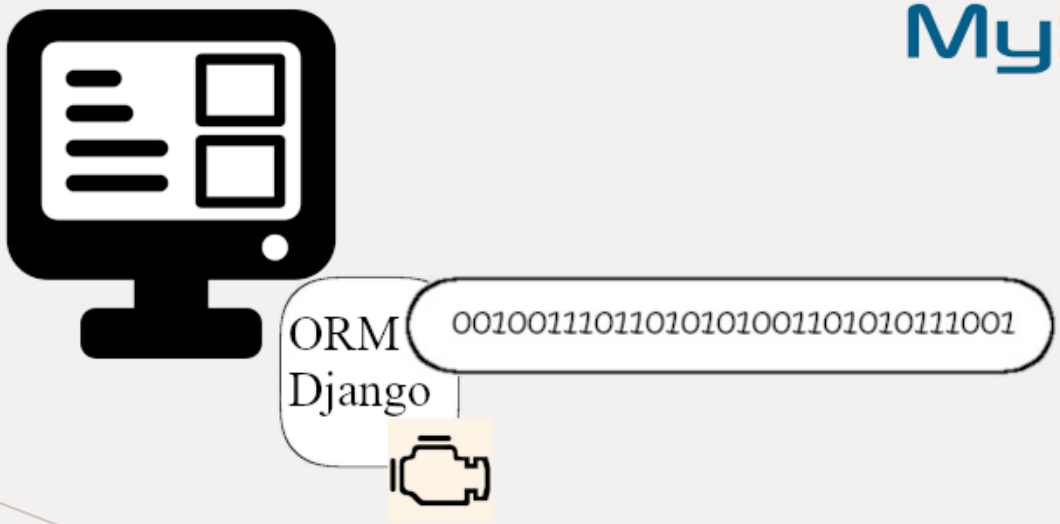

ORM
Django
00100111011010101001101010111001


ORM
Django
00100111011010101001101010111001





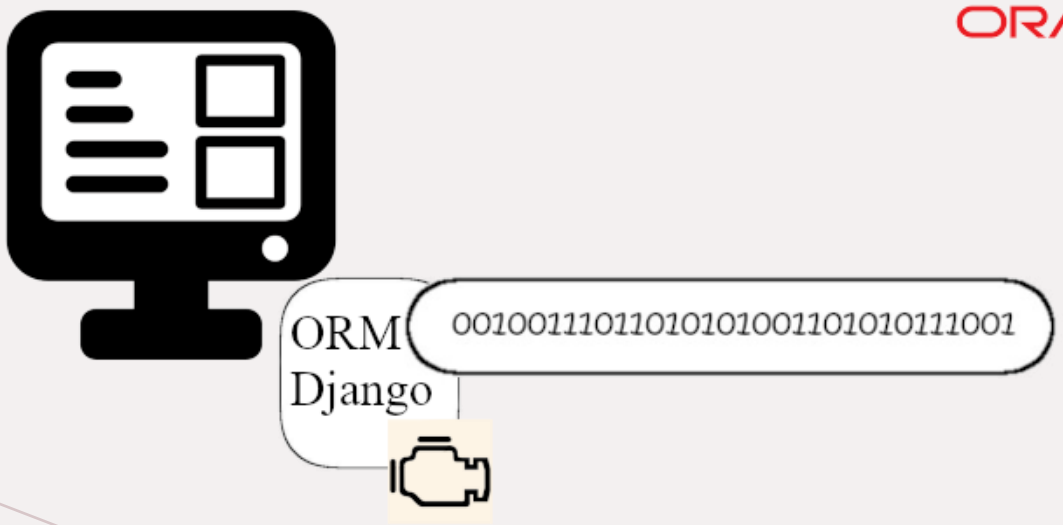
```python
class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
```



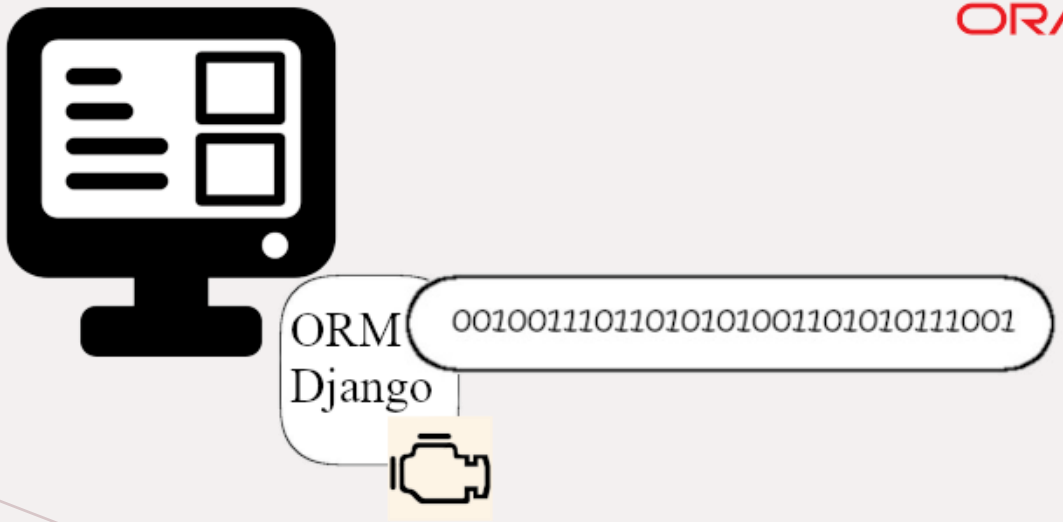


```python
class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
```

```
CREATE TABLE user
(
  user_id INT NOT NULL PRIMARY KEY,
);
```



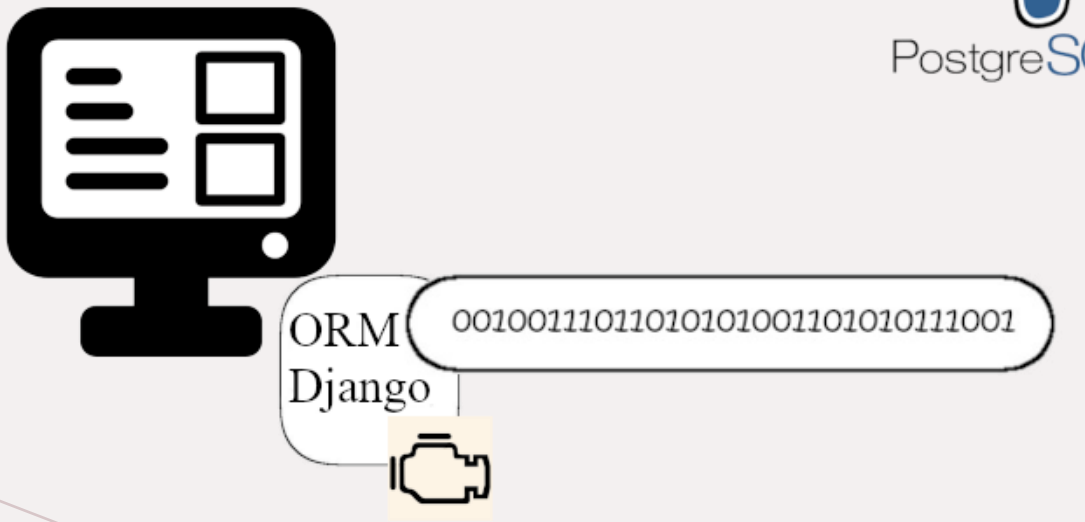

```python
class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
```





```python
class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
```

```
CREATE TABLE user
(
  user_id numeric(10) not null,
  CONSTRAINT user_pk PRIMARY KEY (user_id)
);
```





```python
class User(models.Model):
    user_id = models.IntegerField(primary_key=True)
```

# *PYTHON ORM*

*PRZEGLAD*

SQLAlchemy

THE DATABASE TOOLKIT FOR PYTHON



# The Python SQL Toolkit and Object Relational Mapper

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

## SQLALCHEMY'S PHILOSOPHY

SQL databases behave less like object collections the more size and performance start to matter; object collections behave less like tables and rows the more abstraction starts to matter. SQLAlchemy aims to accommodate both of these principles.

SQLAlchemy

SQLAlchemy considers the database to be a relational algebra engine, not just a collection of tables. Rows can be selected from not only tables but also joins and other select statements; any of these units can be composed into a larger structure. SQLAlchemy's expression language builds on this concept from its core.

SQLAlchemy is most famous for its object-relational mapper (ORM), an optional component that provides the **data mapper pattern**, where classes can be mapped to the database in open ended, multiple ways - allowing the object model and database schema to develop in a cleanly decoupled way from the beginning.

SQLAlchemy's overall approach to these problems is entirely different from that of most other SQL / ORM tools, rooted in a so-called **complimentarity-** oriented approach; instead of hiding away SQL and object relational details behind a wall of automation, all processes are **fully exposed** within a series of composable, transparent tools. The library takes on the job of automating redundant tasks while the developer remains in control of how the database is organized and how SQL is constructed.

The main goal of SQLAlchemy is to change the way you think about databases and SQL!

**Read some key features of SQLAlchemy, as well as what people are saying about SQLAlchemy.**

## CURRENT RELEASES

**1.4.21** - 2021-07-14 - announce
changes | migration notes | docs

**1.3.24** - 2021-03-30 - announce
changes | migration notes | docs



## SPONSOR SQLALCHEMY!

Donate Donate to SQLAlchemy through **PayPal**

Sponsor SQLAlchemy through the Tidelift Subscription

## LATEST NEWS

SQLAlchemy 1.4.21 Released
*Wed, 14 Jul 2021*

SQLAlchemy 1.4.20 Released
*Mon, 28 Jun 2021*

SQLAlchemy 1.4.19 Released


## Project Links

Donate to Pallets
Website
PyPI releases
Source Code
Issue Tracker

## Contents

Flask-SQLAlchemy
- Requirements
- User Guide
- API Reference
- Additional Information

## Quick search

Go

# Flask SQLAlchemy

Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

See the SQLAlchemy documentation to learn how to work with the ORM in depth. The following documentation is a brief overview of the most common tasks, as well as the features specific to Flask-SQLAlchemy.

## Requirements

| Our Version | Python | Flask | SQLAlchemy |
|---|---|---|---|
| 2.x | 2.7, 3.4+ | 0.12+ | 0.8+ or 1.0.10+ w/ Python 3.7 |
| 3.0+ (in dev) | 2.7, 3.5+ | 1.0+ | 1.0+ |



## User Guide

- Quickstart
  - A Minimal Application
  - Simple Relationships
  - Road to Enlightenment
- Introduction into Contexts
- Configuration
  - Configuration Keys
  - Connection URI Format
  - Using custom MetaData and naming conventions
  - Timeouts

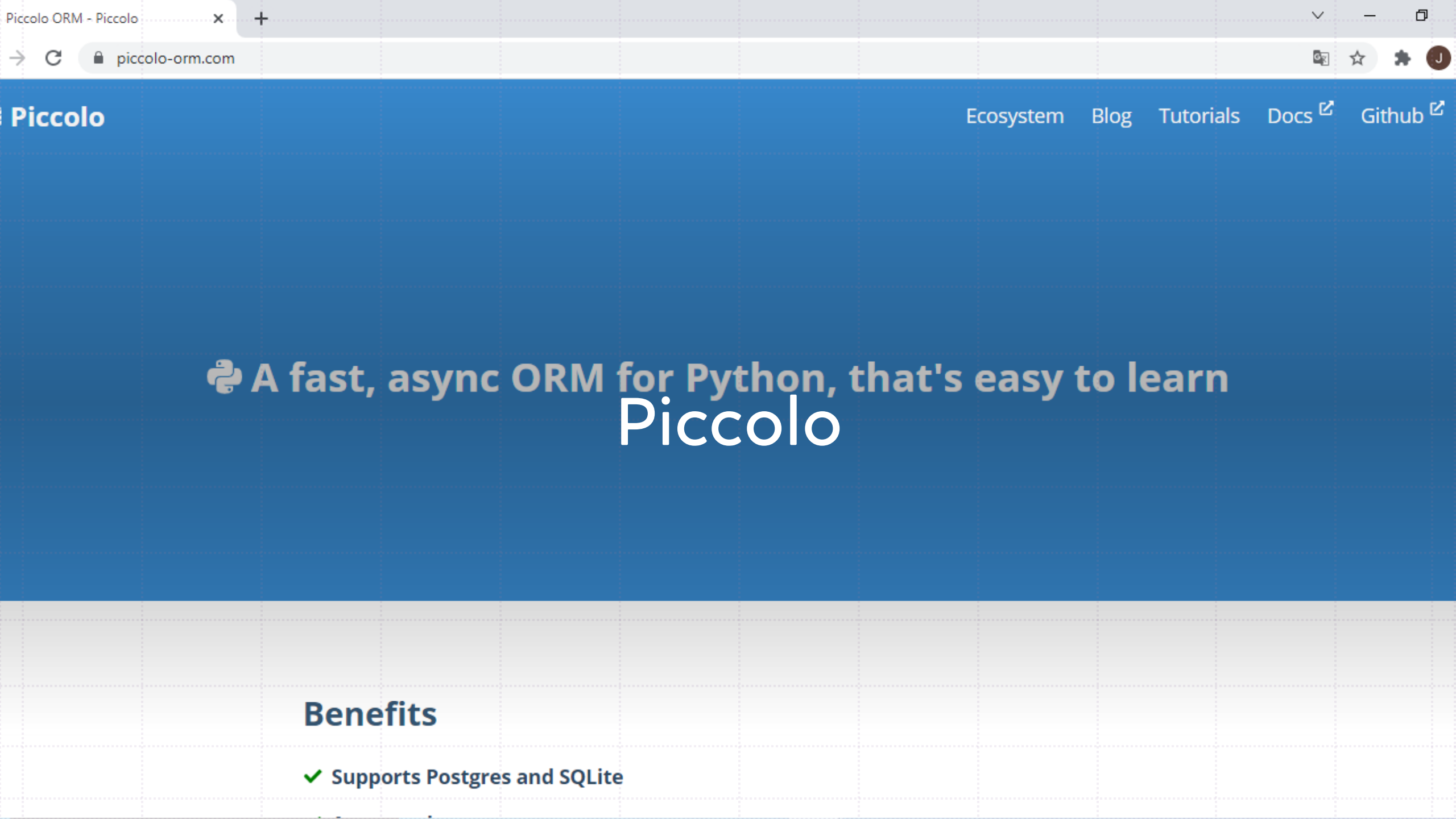
# peewee


Peewee is a simple and small ORM. It has few (but expressive) concepts, making it easy to learn and intuitive to use.

- a small, expressive ORM
- python 2.7+ and 3.4+ (developed with 3.6)
- supports sqlite, mysql, postgresql and cockroachdb
- tons of extensions


Peewee's source code hosted on GitHub.

New to peewee? These may help:


Piccolo ORM - Piccolo
piccolo-orm.com
Piccolo
Ecosystem
Blog
Tutorials
Docs
Github
A fast, async ORM for Python, that's easy to learn
Piccolo
Benefits
Supports Postgres and SQLite

PonyORM - Python ORM with be

https://ponyorm.org

Object-Relational Mapper

Docs Online editor Releases

# Pony is a Python ORM with beautiful query syntax

Write your database queries using Python generators & lambdas

Try PonyORM now Support PonyORM

Free open-source software


Github Twitter Telegram Join the newsletter

## Custom software development

# Making queries

Once you've created your data models, Django automatically gives you a database-abstraction API that lets you create, retrieve, update and delete objects. This document explains how to use this API. Refer to the data model reference for full details of all the various model lookup options.

Throughout this guide (and in the reference), we'll refer to the following models, which comprise a Weblog application:

```
from django.db import models

class Blog(models.Model):
    name = models.CharField(max_length=100)
    tagline = models.TextField()

    def __str__(self):
        return self.name

class Author(models.Model):
    name = models.CharField(max_length=200)
    email = models.EmailField()

    def __str__(self):
        return self.name

class Entry(models.Model):
    blog = models.ForeignKey(Blog, on_delete=models.CASCADE)
    headline = models.CharField(max_length=255)
```

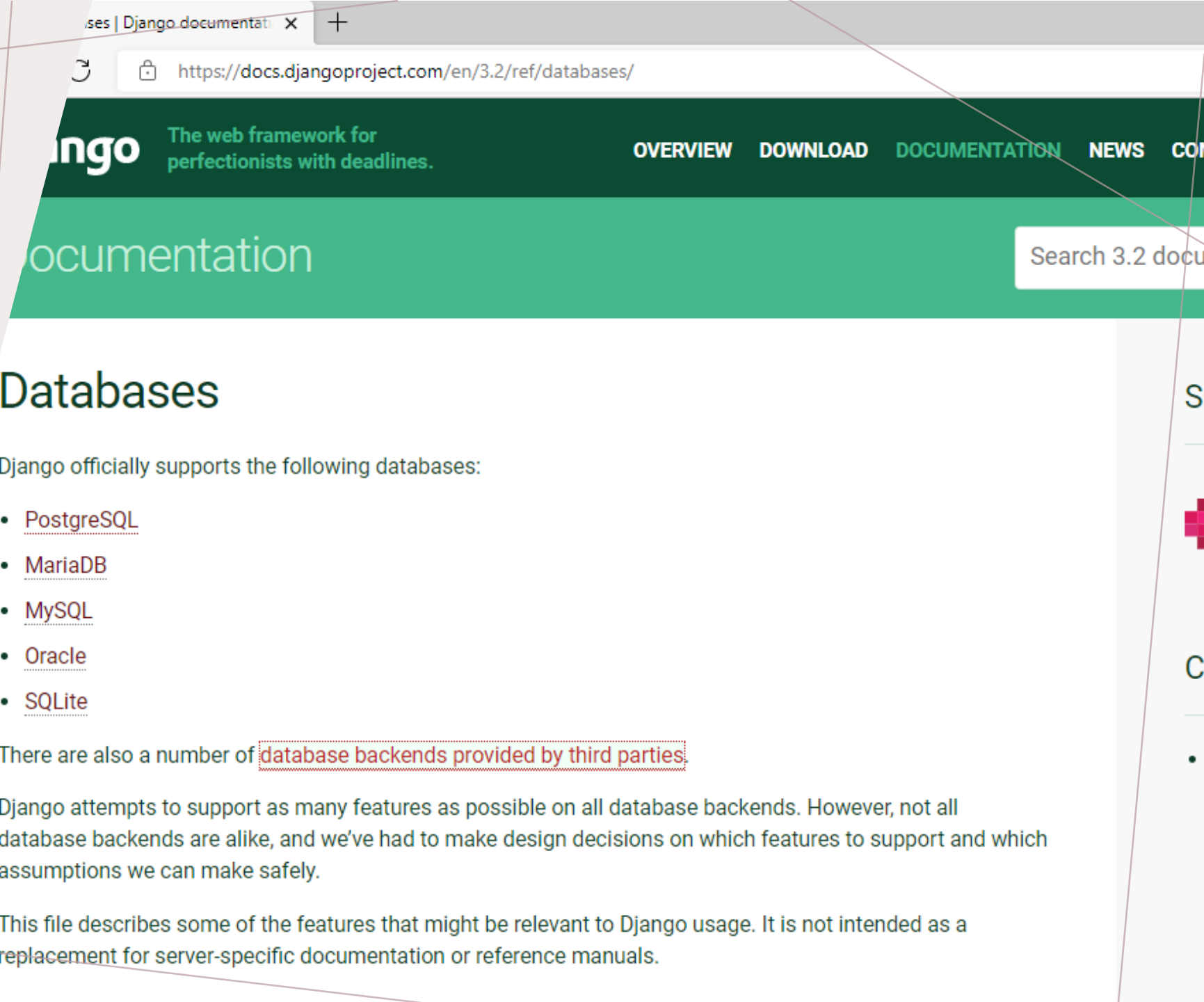




Contents

# *DJANGO ORM*

# OFICJALNE DRIVERY



# *DJANGO ORM*

## *KONFIGURACJA BAZY*

Wszystkie informacje potrzebne do połączenia się z bazą danych znajdują się w pliku **settings.py** w zmiennej **DATABASES**

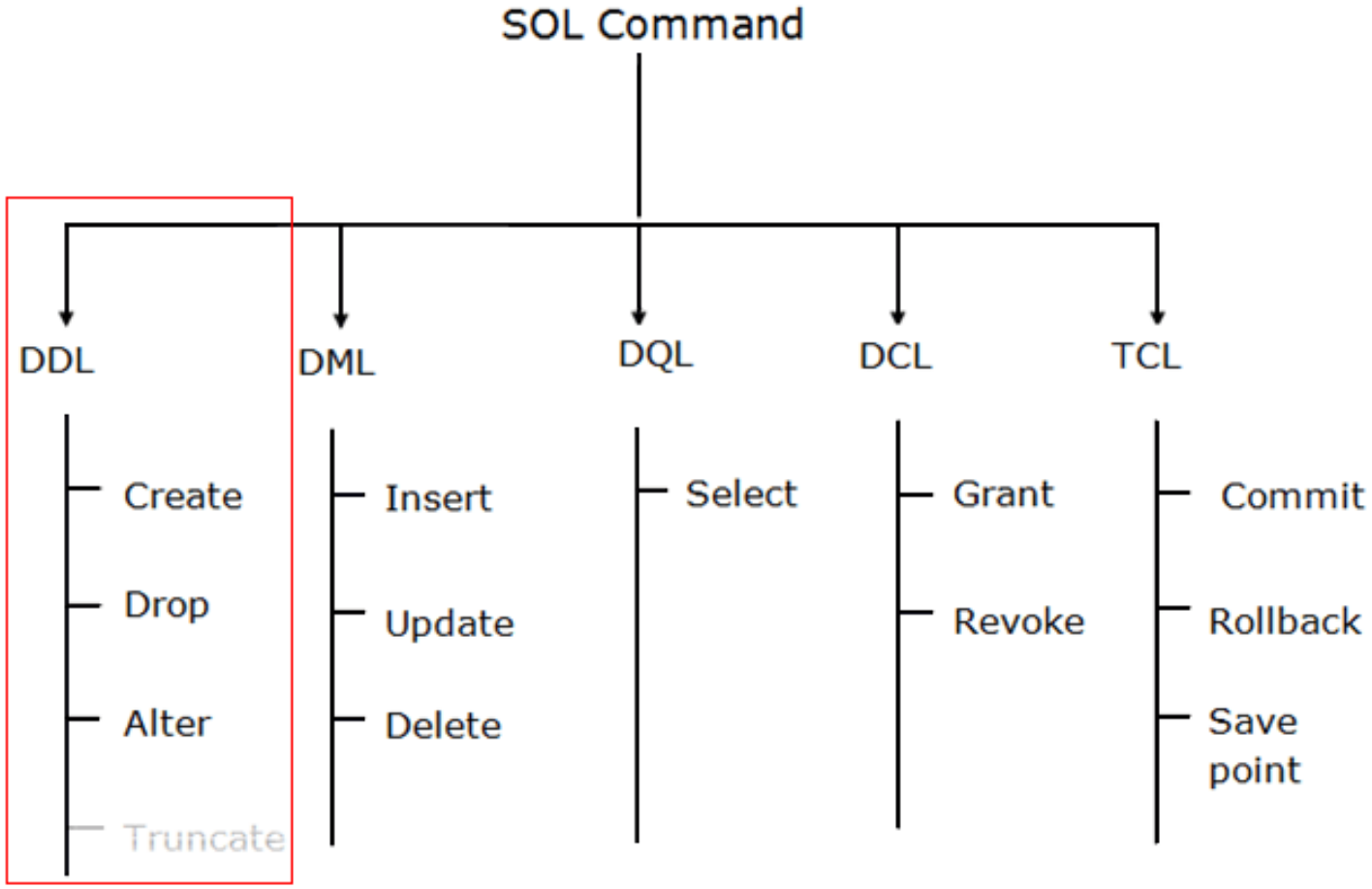
Przykładową konfigurację dla baz **mysql** oraz **postgresql** można znaleźć w dodatkach w rozdziale Konfiguracja połączenia z bazą. W ramach prezentacji pracować będziemy z bazą **sqlite**, ponieważ jest to najprostszy do skonfigurowania typ bazy, a jednocześnie jest to domyślnie skonfigurowana baza we frameworku Django. Należy mieć jednak na uwadzę, że bazy sqlite nie są odpowiednim rozwiązaniem dla środowisk produkcyjnych.

# *DJANGO ORM*

*INSTRUKCJE DDL*

# DDL – DATA DEFINITION LANGUAGE

- **DDL** – Data Definition Language
- **DML** – Data Manipulation Language
- **DQL** – Data Query Language
- **DCL** – Data Control Language
- **TCL** – Transaction Control Language



# *INSTRUKCJE DDL*

*SYSTEM MIGRACJI*

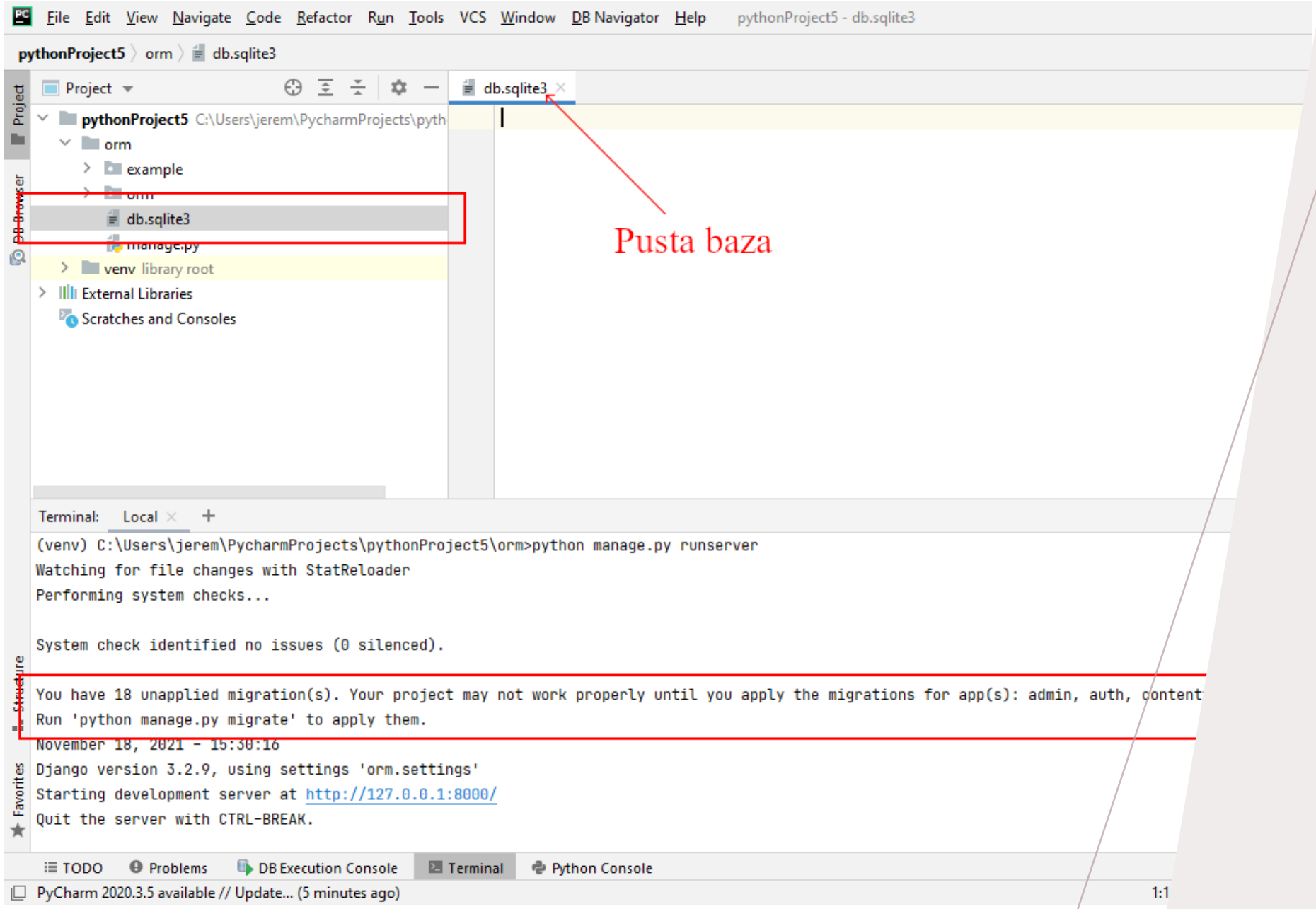
# *MIGRACJE*

Django jest wyposażony w system migracji. Migracje to zmiany struktury bazy danych (create/alter/drop table) odzwierciedlające zmiany wprowadzone w kodzie. Wszystkie instrukcje ddl są realizowane na bazie za pomocą migracji.

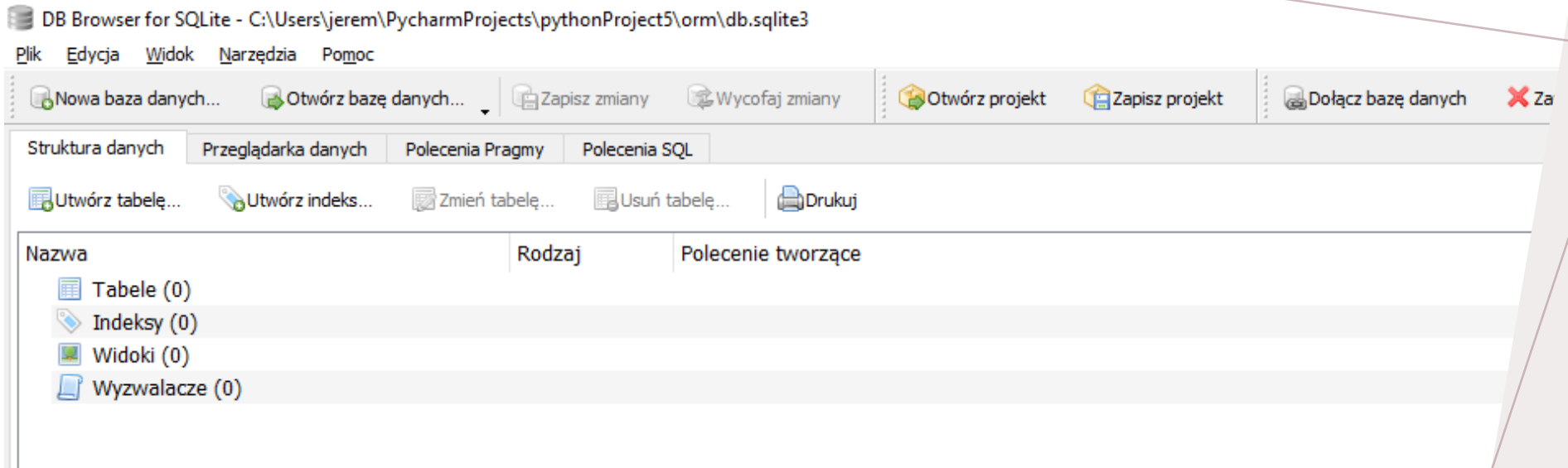
# *MIGRACJE POCZATKOWE*

Nowy projekt Django  wymaga zastosowania początkowych migracji. Migracje początkowe odzwierciedlają zestaw tabel potrzebnych domyślnym aplikacjom Django do działania. Po zastosowaniu migracji baza danych zawiera wszystkie potrzebne tabele. Bez zastosowania początkowych migracji projekt nie będzie działać prawidłowo, ponieważ część funkcji (np. walidacja CSRF) wymaga stworzenia odpowiednich tabel w bazie.

Początkowe migracje są przygotowane przez twórców Django. Tworząc nowy projekt Django naszym zadaniem jest dopilnować, aby początkowe migracje zostały wykonane na bazie.

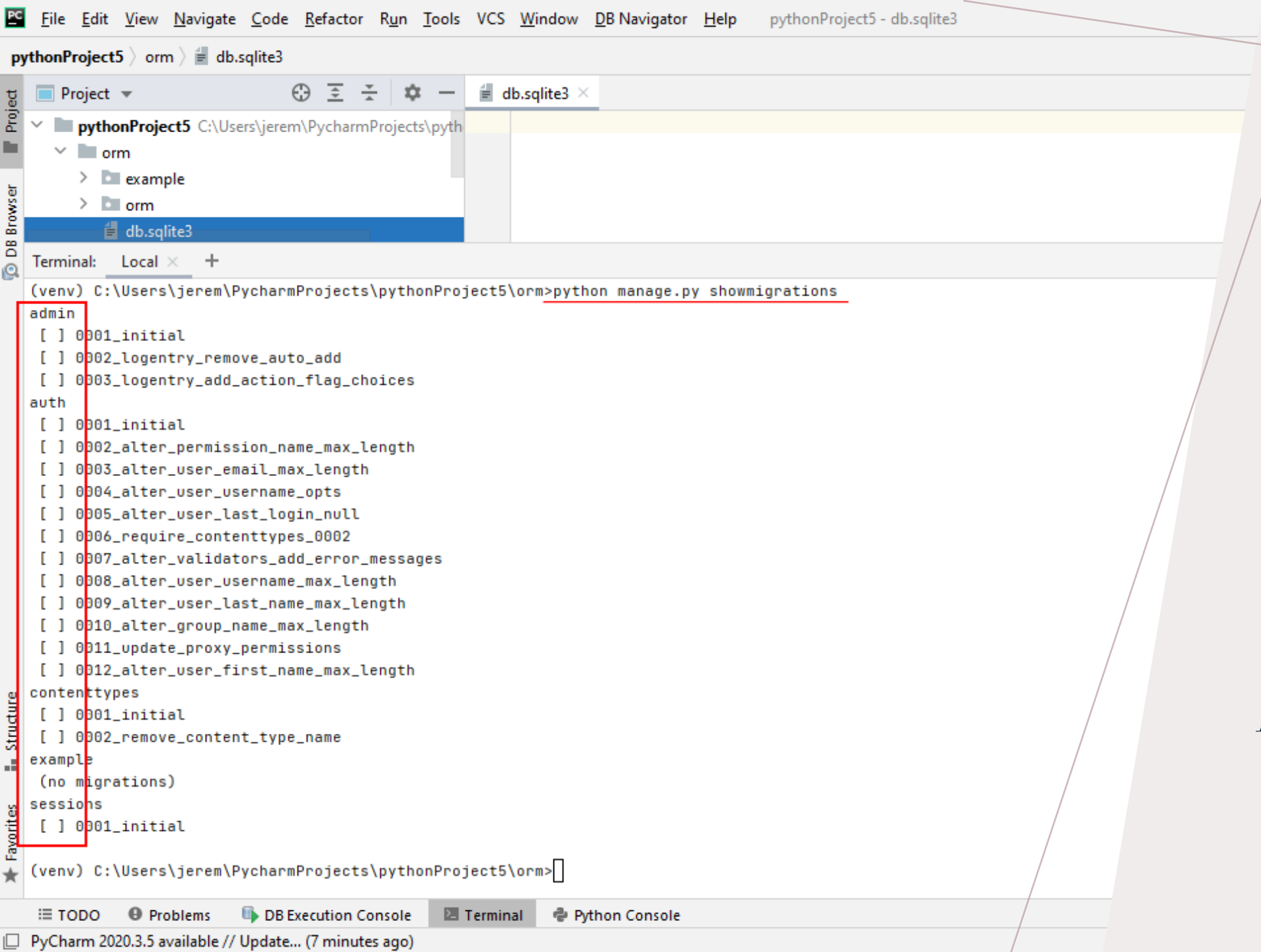



# POCZĄTKOWE MIGRACJE



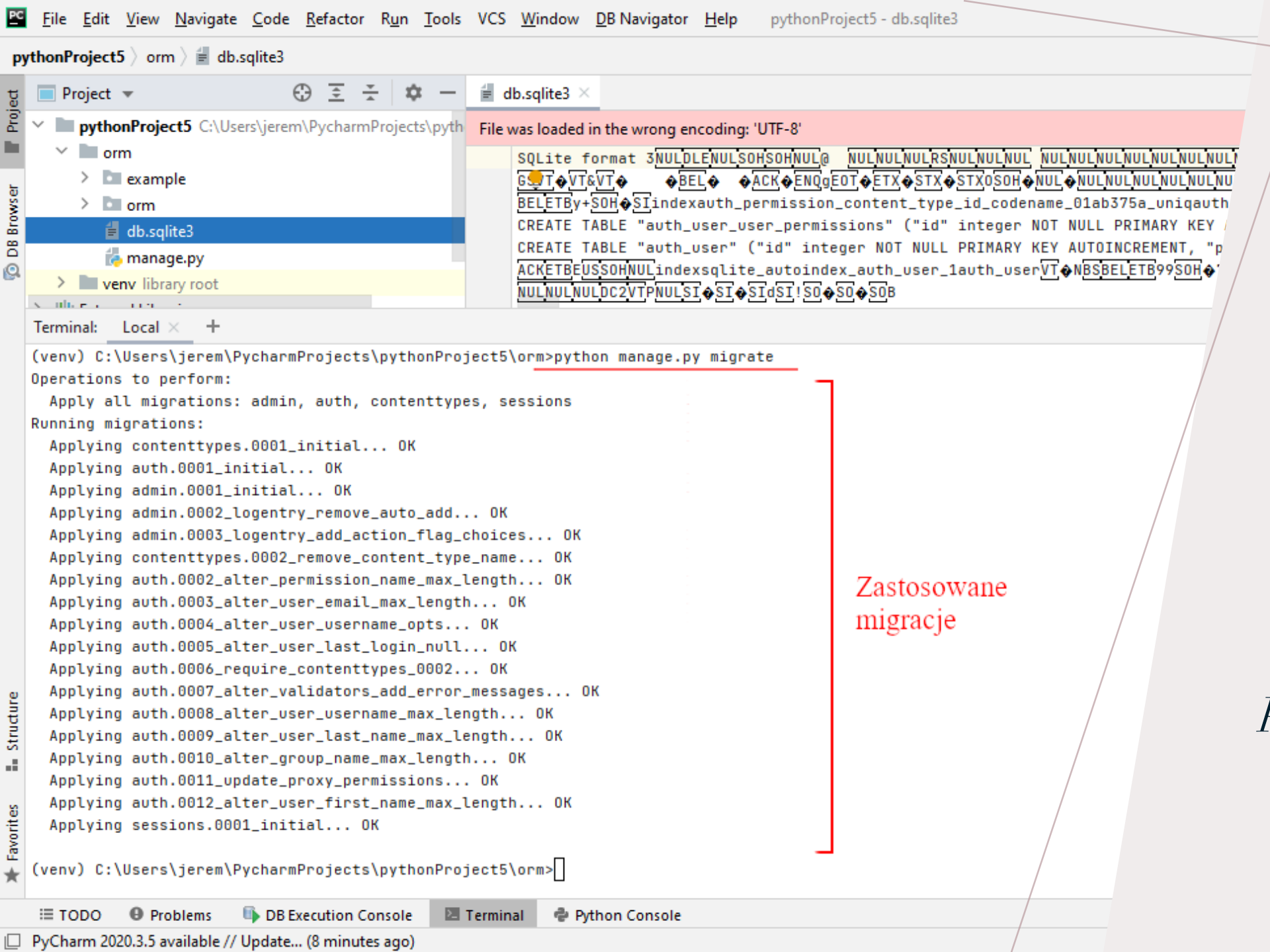

# POCZĄTKOWE MIGRACJE

## PUSTA BAZA




# POCZĄTKOWE MIGRACJE

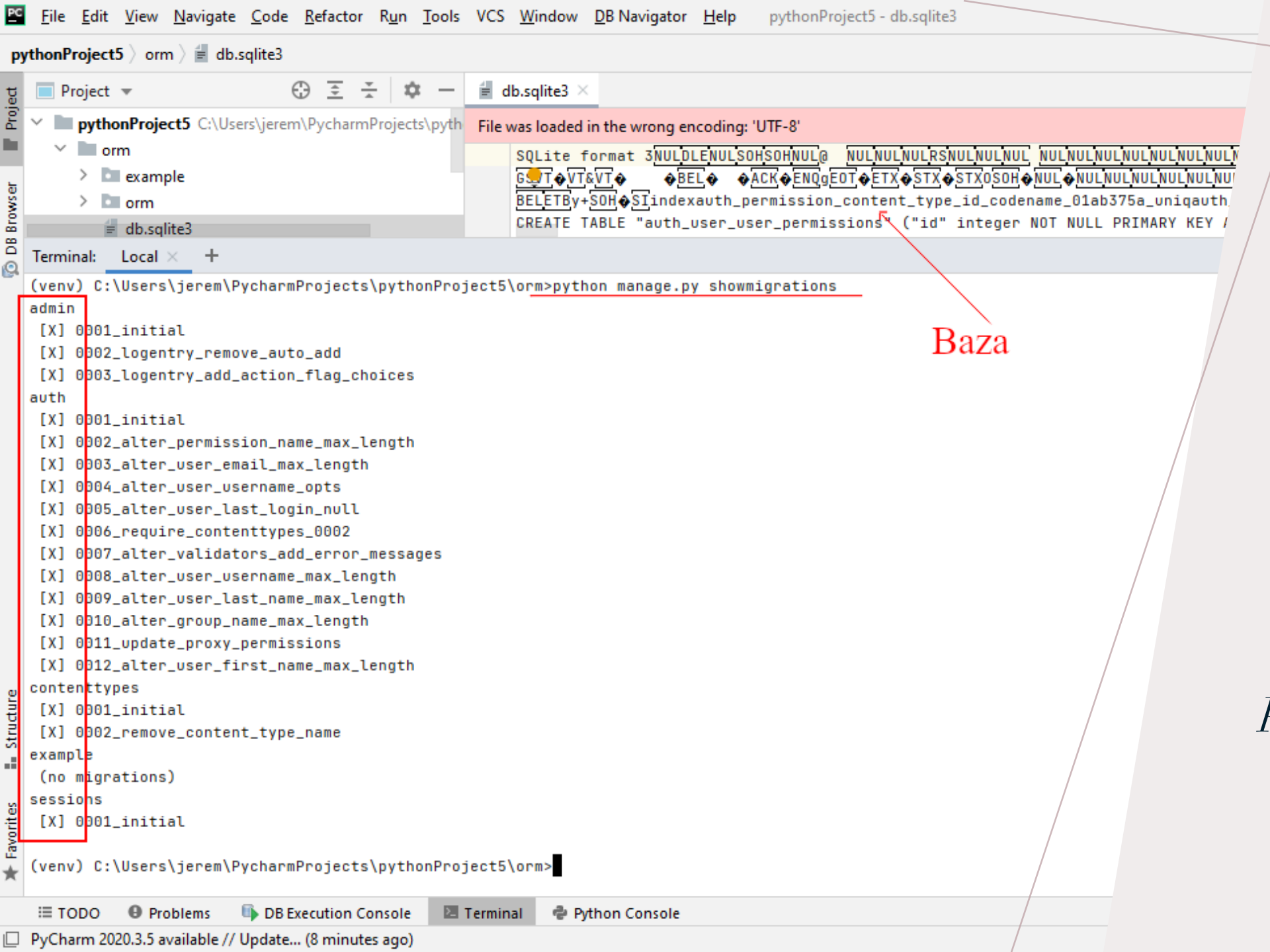
## SHOWMIGRATIONS




Zastosowane migracje

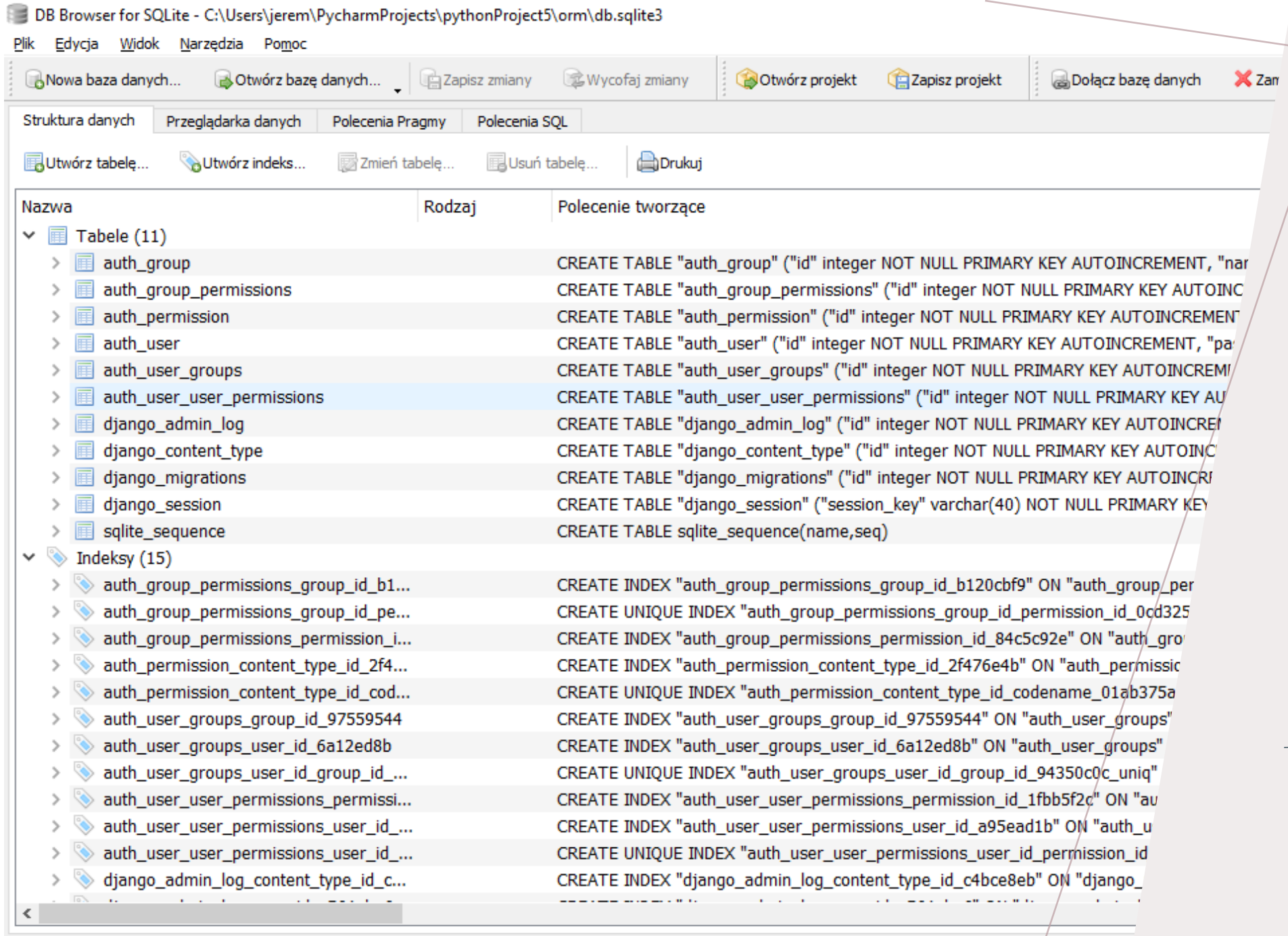
# POCZĄTKOWE MIGRACJE

## *MIGRATE*




# POCZĄTKOWE MIGRACJE

## SHOWMIGRATIONS




# POCZĄTKOWE MIGRACJE

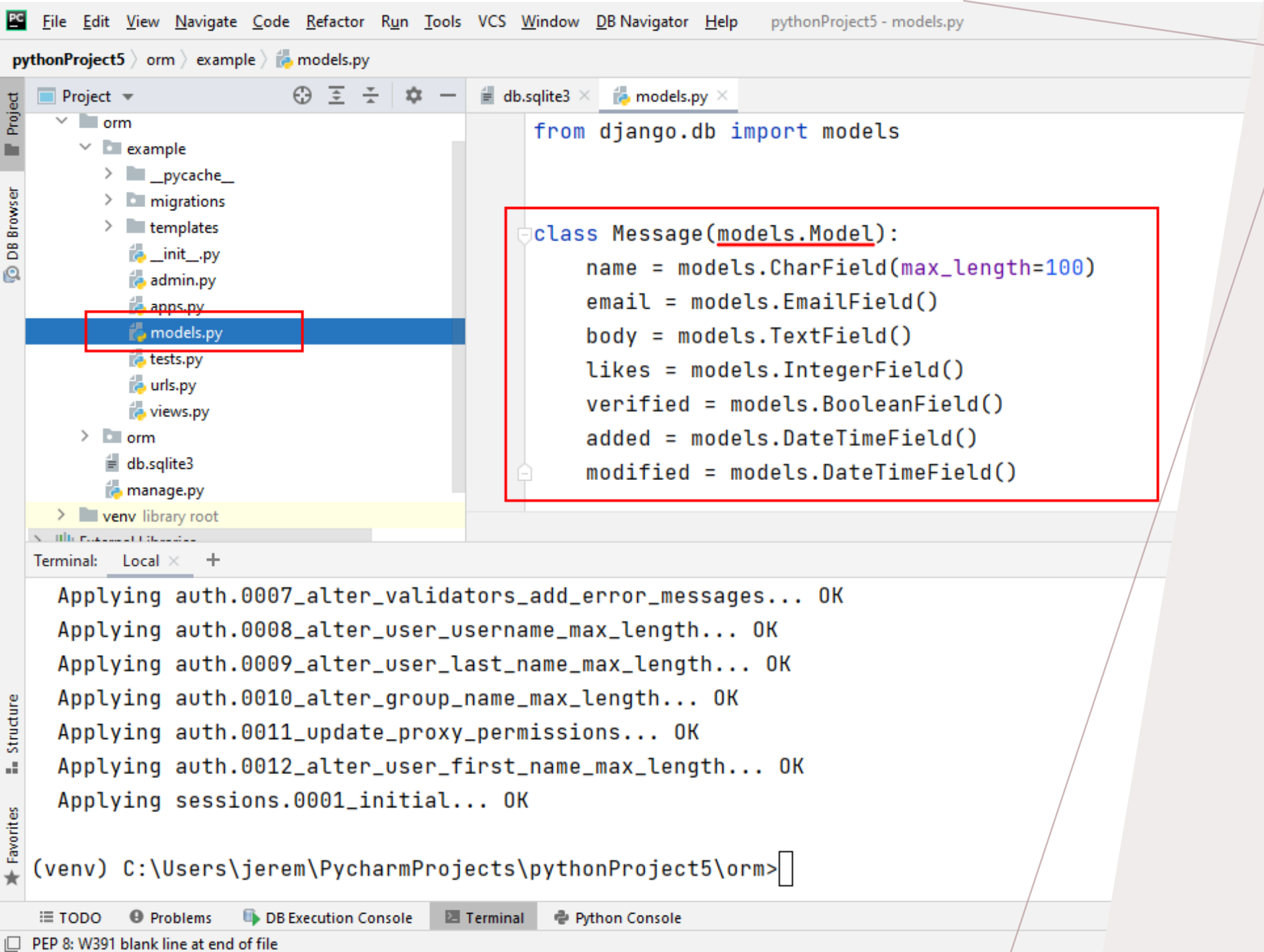
## BAZA

# *INSTRUKCJE DDL*

*MODELE*

# *MODELE*

Model to klasa dziedzicząca po klasie **django.db.models.Model**. Jest obiektową reprezentacją tabeli z bazy danych. Silnik Django szuka modeli (klas dziedziczących po klasie django.db.models.Model) w pliku models.py każdej zarejestrowanej aplikacji.




# DEFINICJA MODELU

# *INSTRUKCJE DDL*

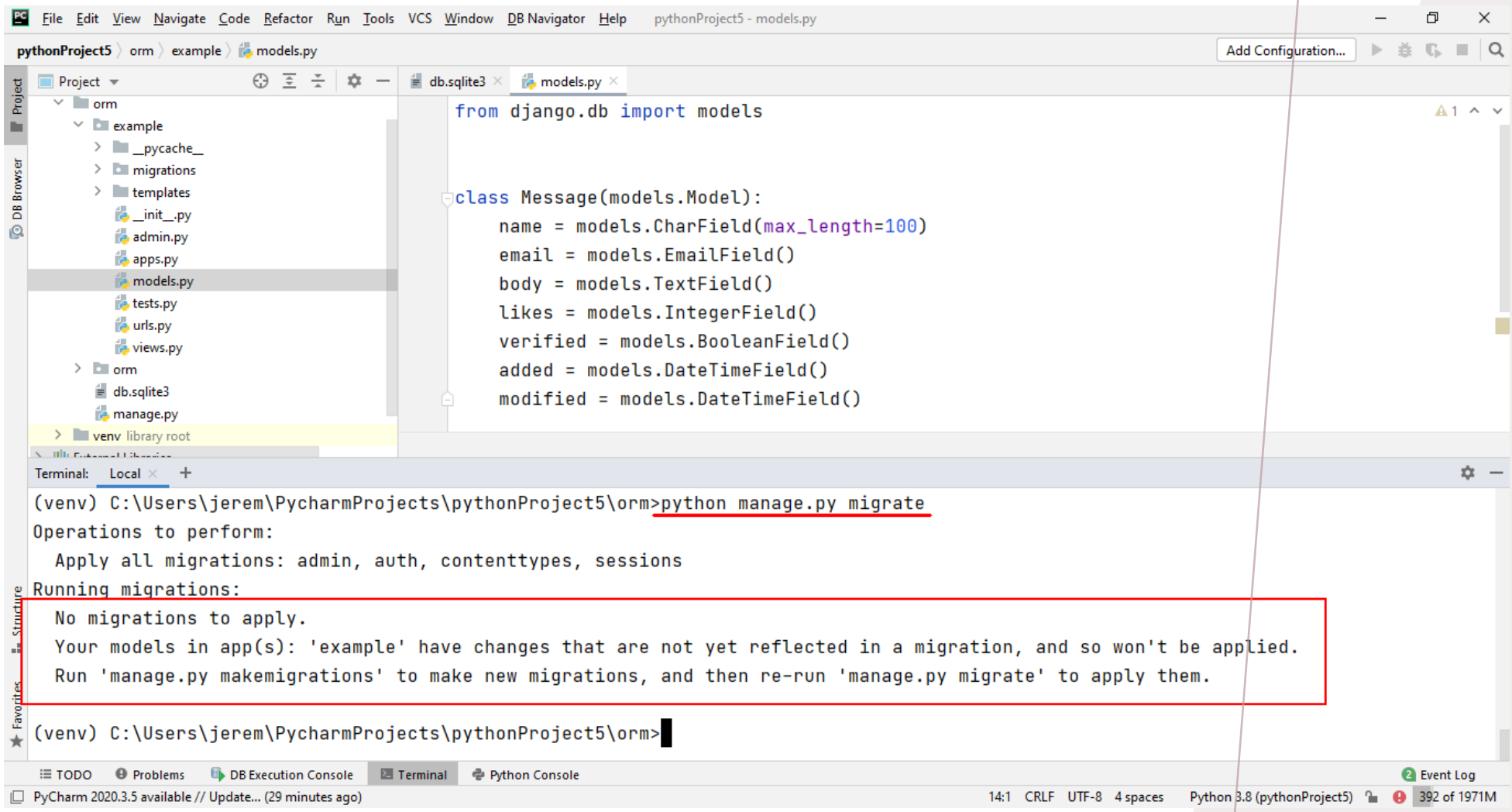
## *MODELE I SYSTEM MIGRACJI*

# *MODELE I SYSTEM MIGRACJI*

Silnik Django śledzi zmiany w plikach models.py każdej zarejestrowanej aplikacji. Żeby wprowadzona w pliku models.py zmiana znalazła odzwierciedlenie w bazie danych należy przygotować dla tej zmiany migracje (komenda **makemigrations**), a następnie przygotowaną migracje zastosować na bazie (komenda **migrate**).

Uwaga!

Zasada ta dotyczy tylko tych zmian w modelu, które mają wpływ na strukturę tabeli bazodanowej reprezentowanej przez model, czyli **zmian dotyczących kolumn tabeli lub samej tabeli.** Jeżeli zmodyfikujemy model poprzez wprowadzenie zmiany, która nie będzie miała wpływu na struktrę tabeli (np. dodamy reprezentację napisową, czyli metodę __str__) to taka zmiana modelu nie wygeneruje migracji.

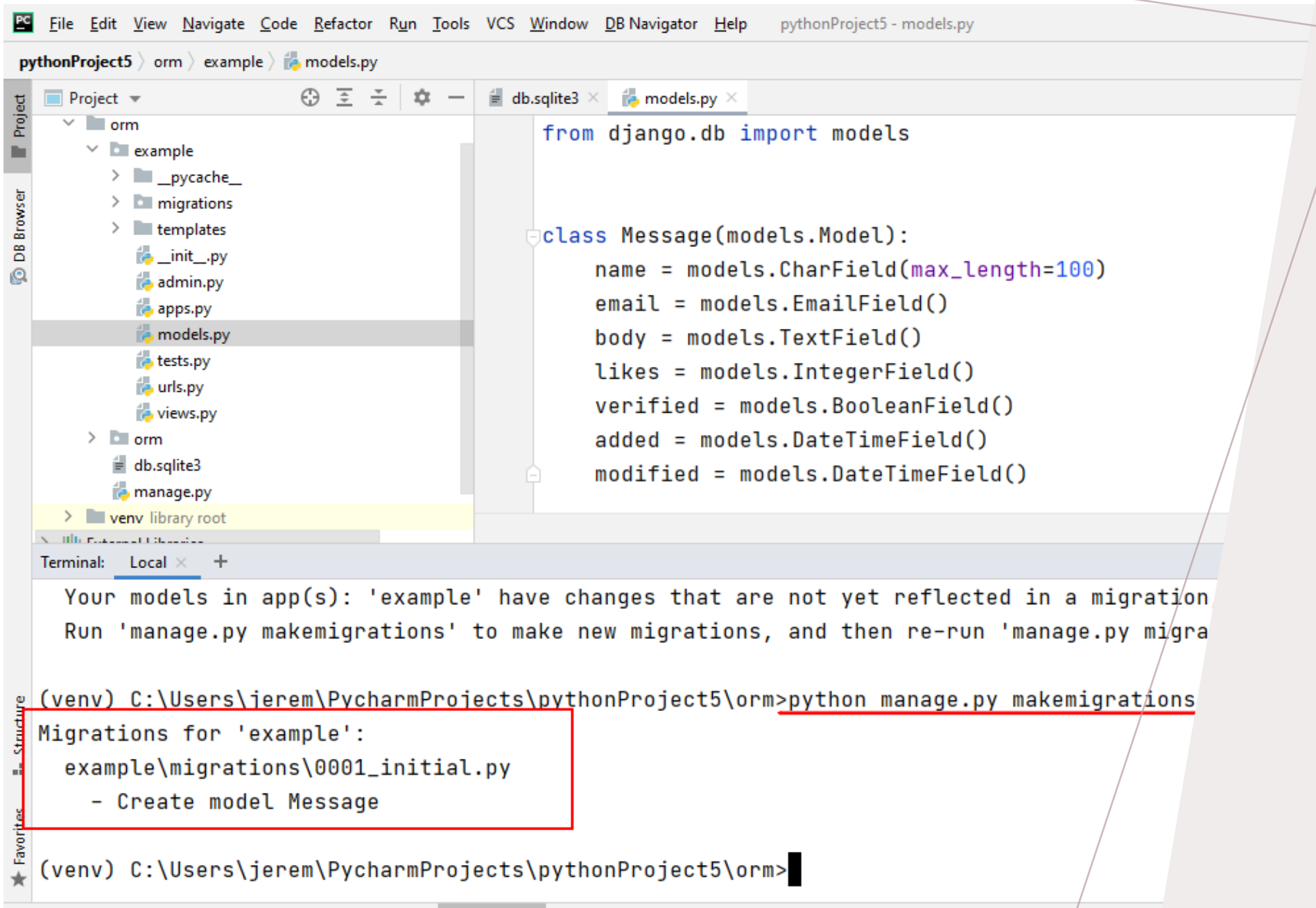



*TYM RAZEM SAMA KOMENDA MIGRATE NIE WYSTARCZY*

# *MODELE I SYSTEM MIGRACJI*

Do rzutowania modelu na tabelę w bazie danych wykorzystywane są dwie komendy:

- python manage.py **makemigrations** – przygotowuje migracje do zastosowania. Komenda generuje plik z kodem migracji w folderze migrations znajdującym się w katalogu aplikacji
- python manage.py **migrate** – stosuje przygotowane migracje na bazie. Komenda wykonuje przygotowane pliki z kodem migracji (w tym miejscu następuje fizyczne połączenie z bazą i wprowadzenie w niej zmian)

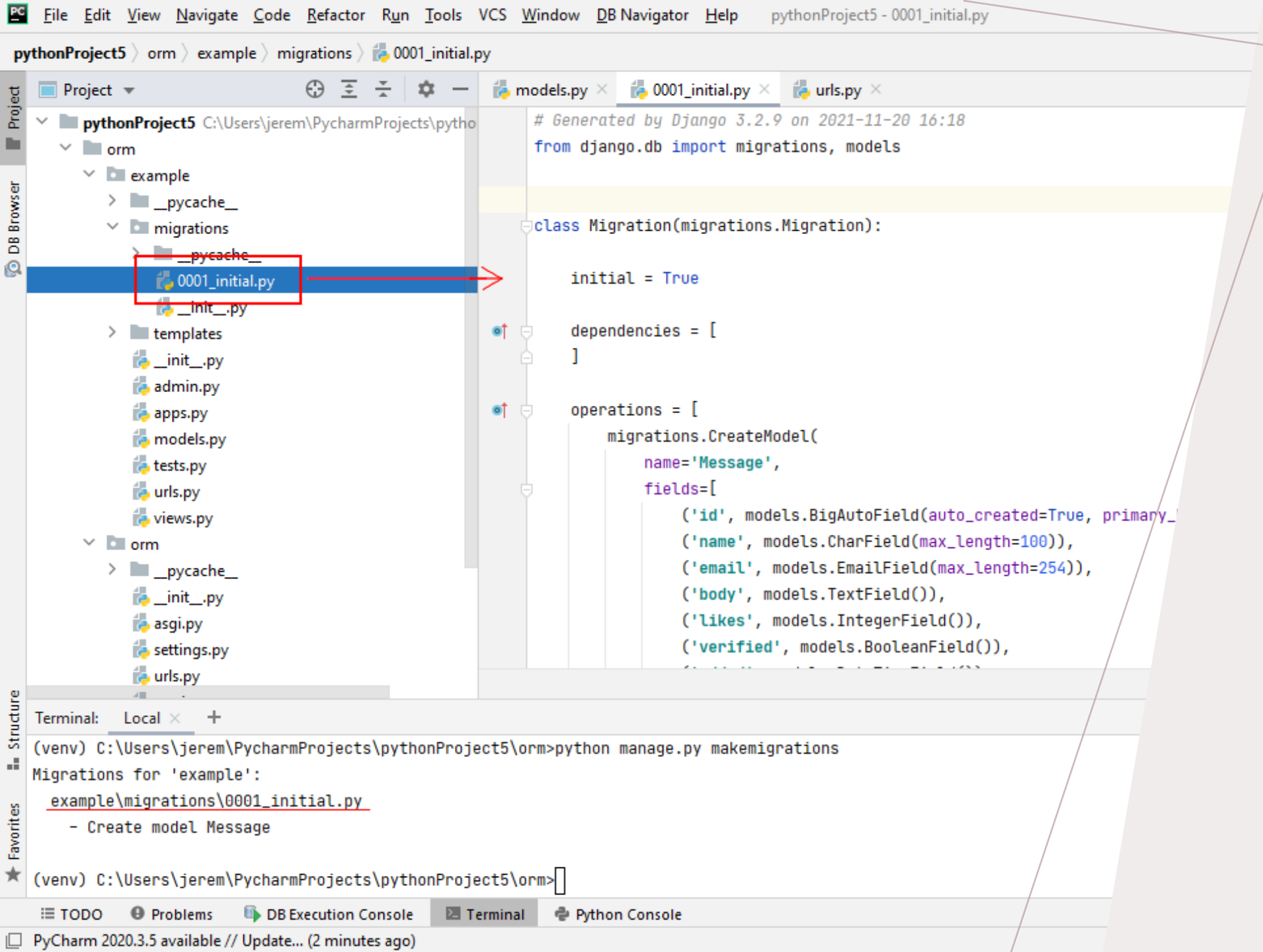



# MODELE

## MAKEMIGRATIONS

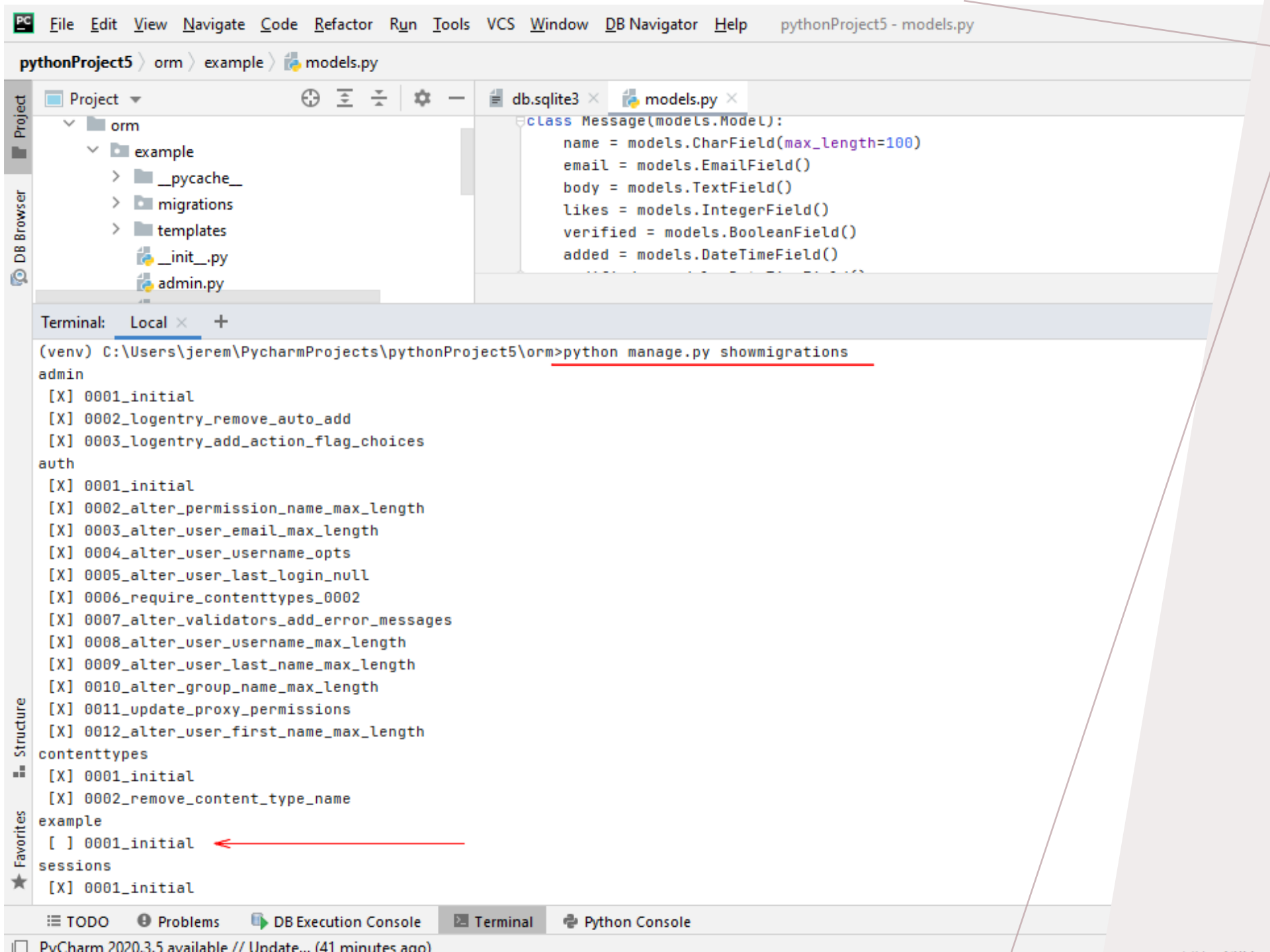
# *MODELE I SYSTEM MIGRACJI*

Po przygotowaniu migracji jej kod przechowywany jest w pliku w postaci skryptu Python. Każda migracja posiada odpowiadający jej skrypt.



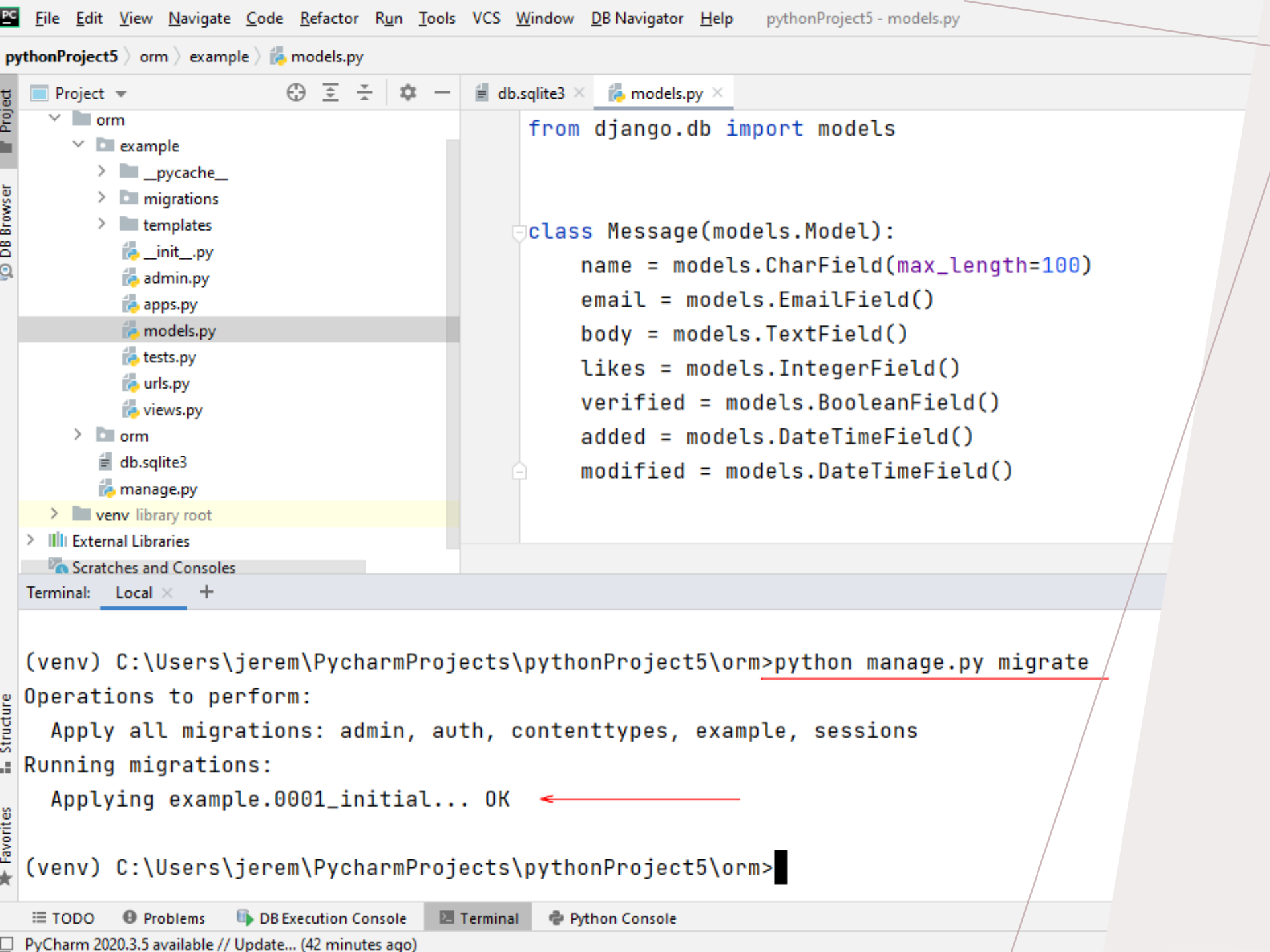

# MODELE

## SKRYPT Z MIGRACJA



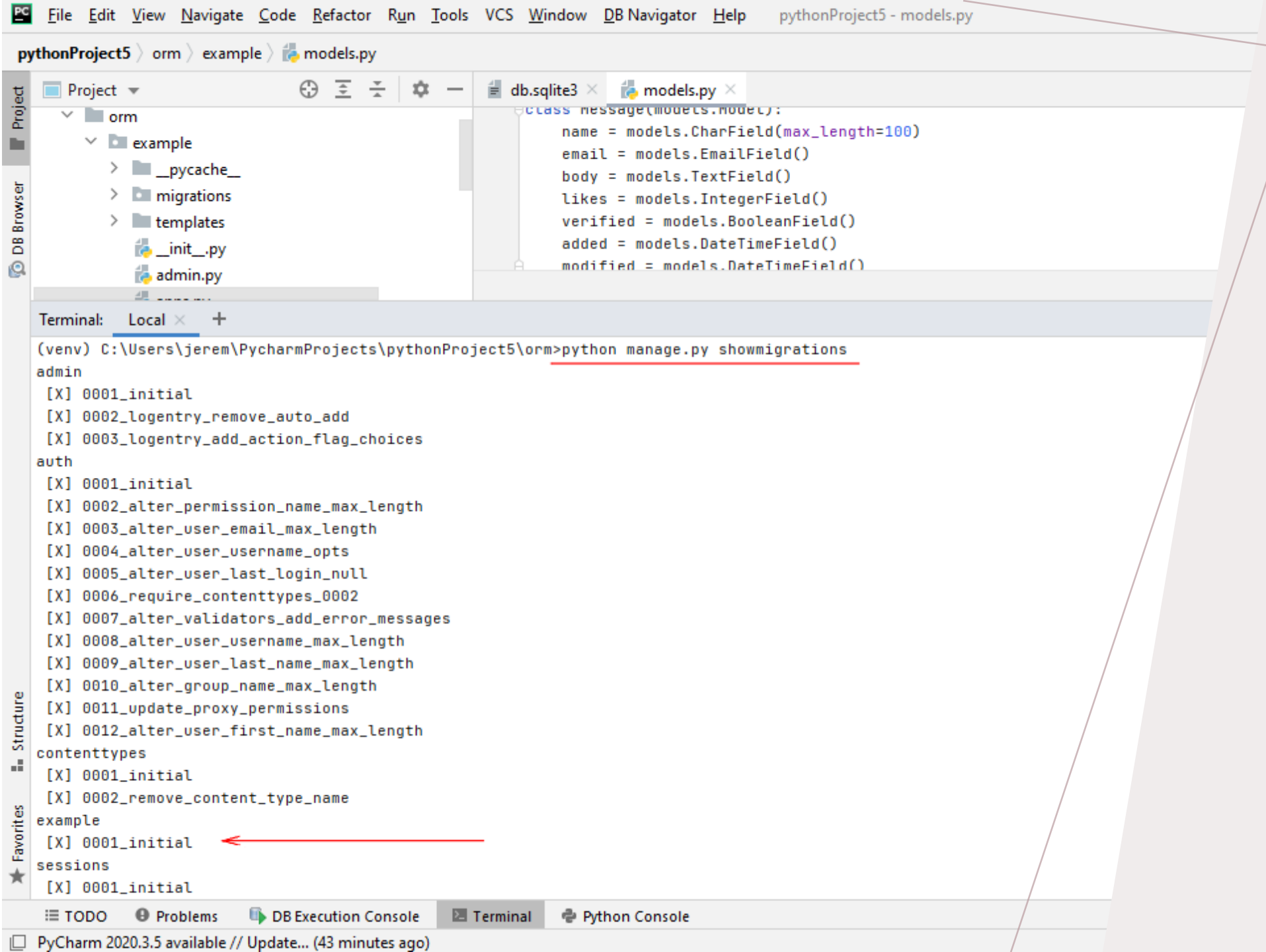

# MODELE

## SHOWMIGRATIONS



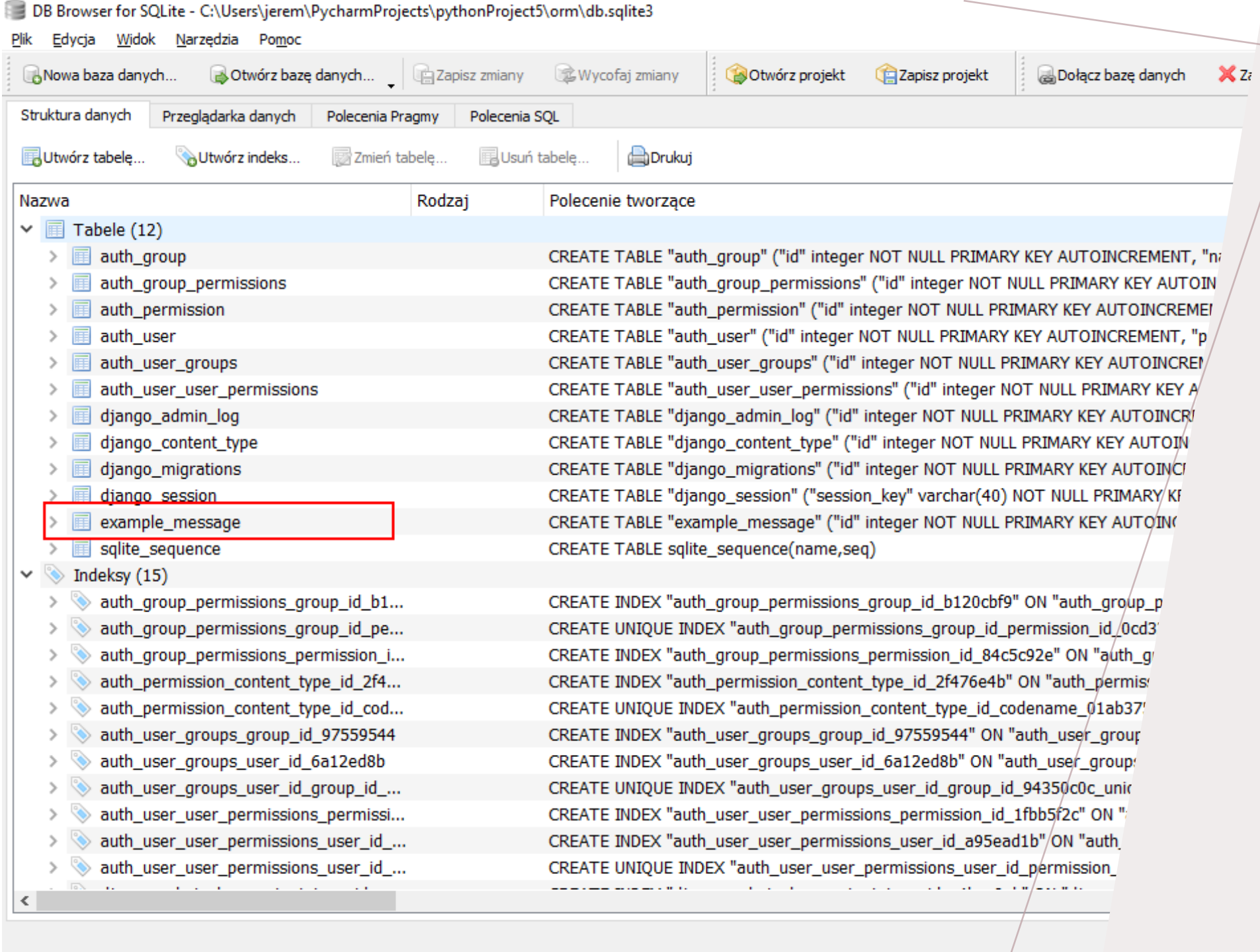

# MODELE

## MIGRATE




# MODELE

## SHOWMIGRATIONS

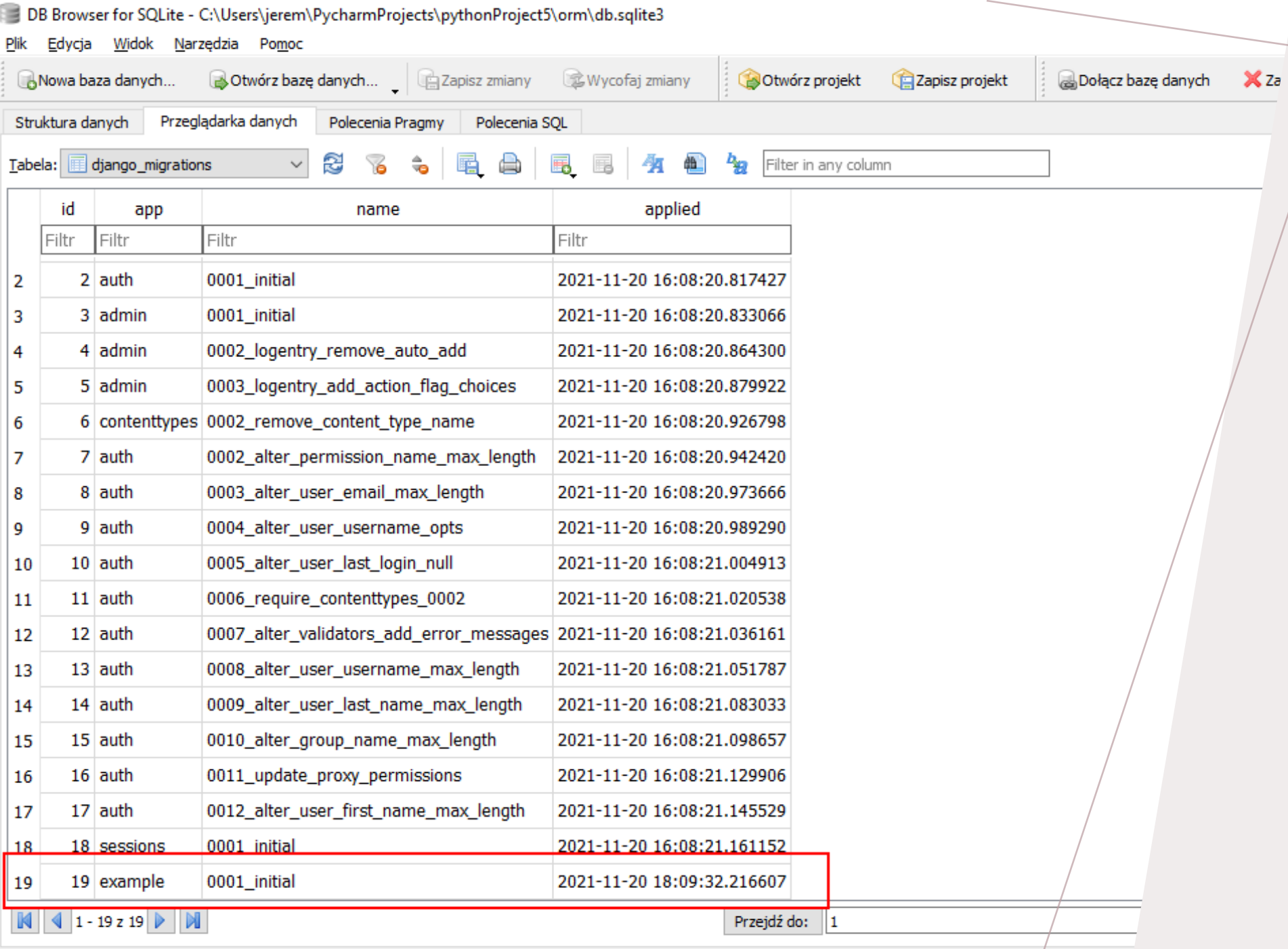

DB Browser for SQLite - C:\Users\jerem\PycharmProjects\pythonProject5\orm\db.sqlite3
Plik Edycja Widok Narzędzia Pomoc
Nowa baza danych... Otwórz bazę danych... Zapisz zmiany Wycofaj zmiany Otwórz projekt Zapisz projekt Dołącz bazę danych
Struktura danych Przeglądarka danych Polecenia Pragmy Polecenia SQL
Utwórz tabelę... Utwórz indeks... Zmień tabelę... Usuń tabelę... Drukuj
Nazwa Rodzaj Polecenie tworzące
Tabele (12)
auth_group CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "n
auth_group_permissions CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOIN
auth_permission CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREME
auth_user CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "p
auth_user_groups CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCRE
auth_user_user_permissions CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY A
django_admin_log CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCR
django_content_type CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOIN
django_migrations CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINC
django_session CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY K
example_message CREATE TABLE "example_message" ("id" integer NOT NULL PRIMARY KEY AUTOIN
sqlite_sequence CREATE TABLE sqlite_sequence(name,seq)
Indeksy (15)
auth_group_permissions_group_id_b1... CREATE INDEX "auth_group_permissions_group_id_b120cbf9" ON "auth_group_p
auth_group_permissions_group_id_pe... CREATE UNIQUE INDEX "auth_group_permissions_group_id_permission_id_0cd3
auth_group_permissions_permission_i... CREATE INDEX "auth_group_permissions_permission_id_84c5c92e" ON "auth_g
auth_permission_content_type_id_2f4... CREATE INDEX "auth_permission_content_type_id_2f476e4b" ON "auth_permis
auth_permission_content_type_id_cod... CREATE UNIQUE INDEX "auth_permission_content_type_id_codename_01ab37
auth_user_groups_group_id_97559544 CREATE INDEX "auth_user_groups_group_id_97559544" ON "auth_user_group
auth_user_groups_user_id_6a12ed8b CREATE INDEX "auth_user_groups_user_id_6a12ed8b" ON "auth_user_group
auth_user_groups_user_id_group_id_... CREATE UNIQUE INDEX "auth_user_groups_user_id_group_id_94350c0c_uni
auth_user_user_permissions_permissi... CREATE INDEX "auth_user_user_permissions_permission_id_1fbb5f2c" ON "
auth_user_user_permissions_user_id_... CREATE INDEX "auth_user_user_permissions_user_id_a95ead1b" ON "auth_
auth_user_user_permissions_user_id_... CREATE UNIQUE INDEX "auth_user_user_permissions_user_id_permission_

# MODELE

## BAZA

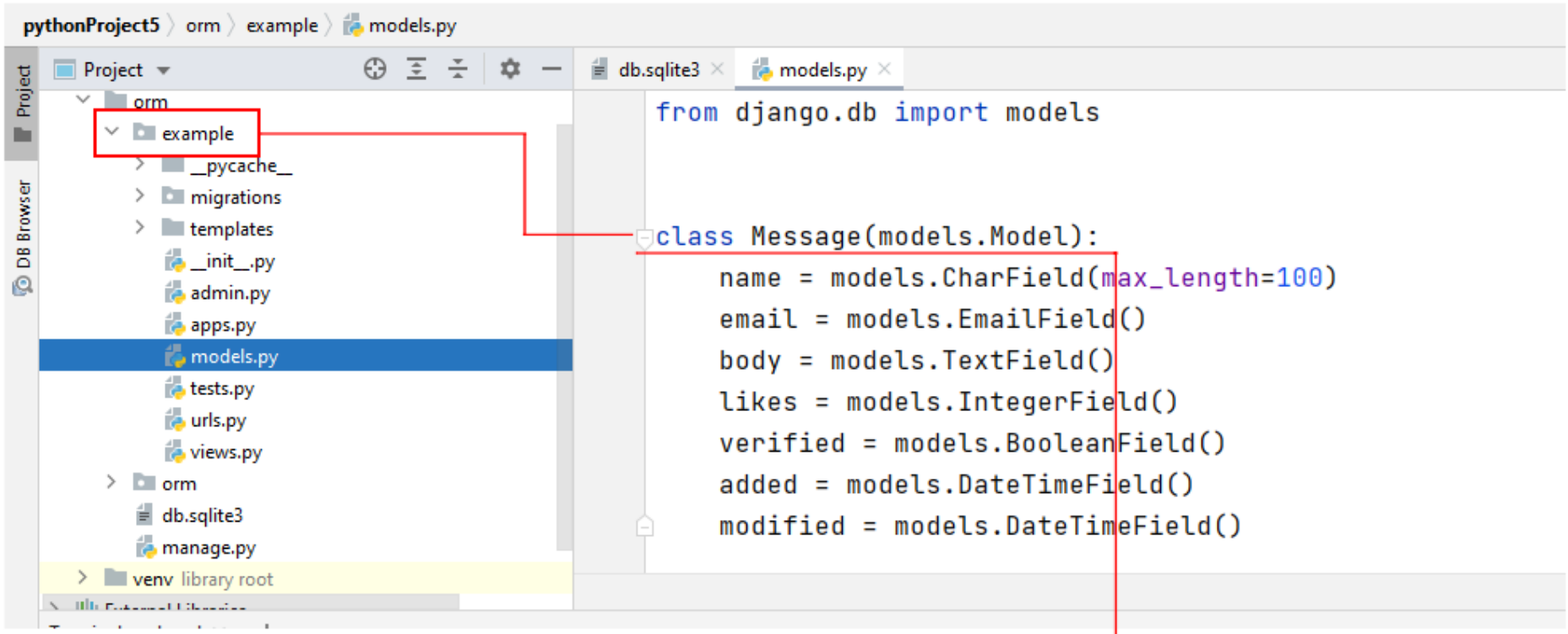
# *MODELE I SYSTEM MIGRACJI*

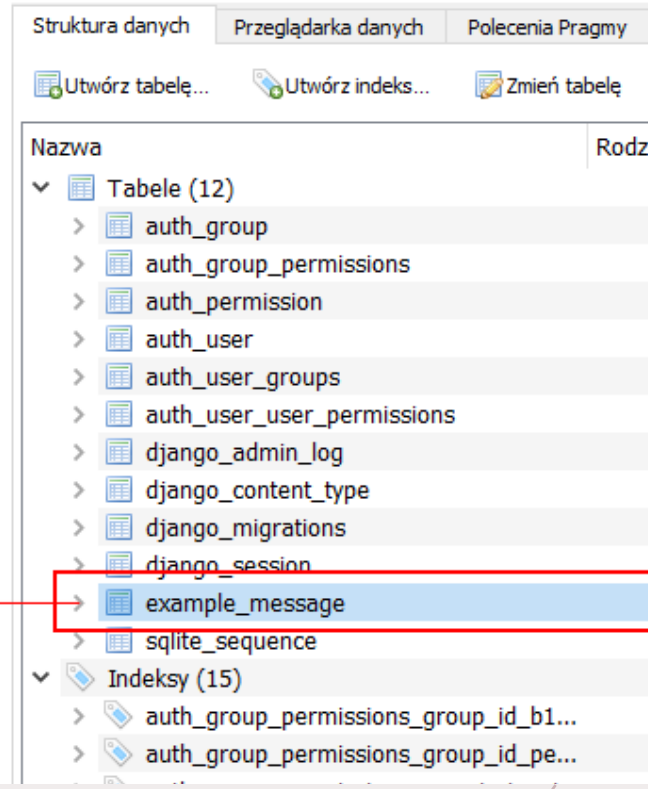
Po wykonaniu migracji wpis informujący o wykonaniu danej migracji odkładany jest w bazie w tabeli django_migrations.




# MODELE

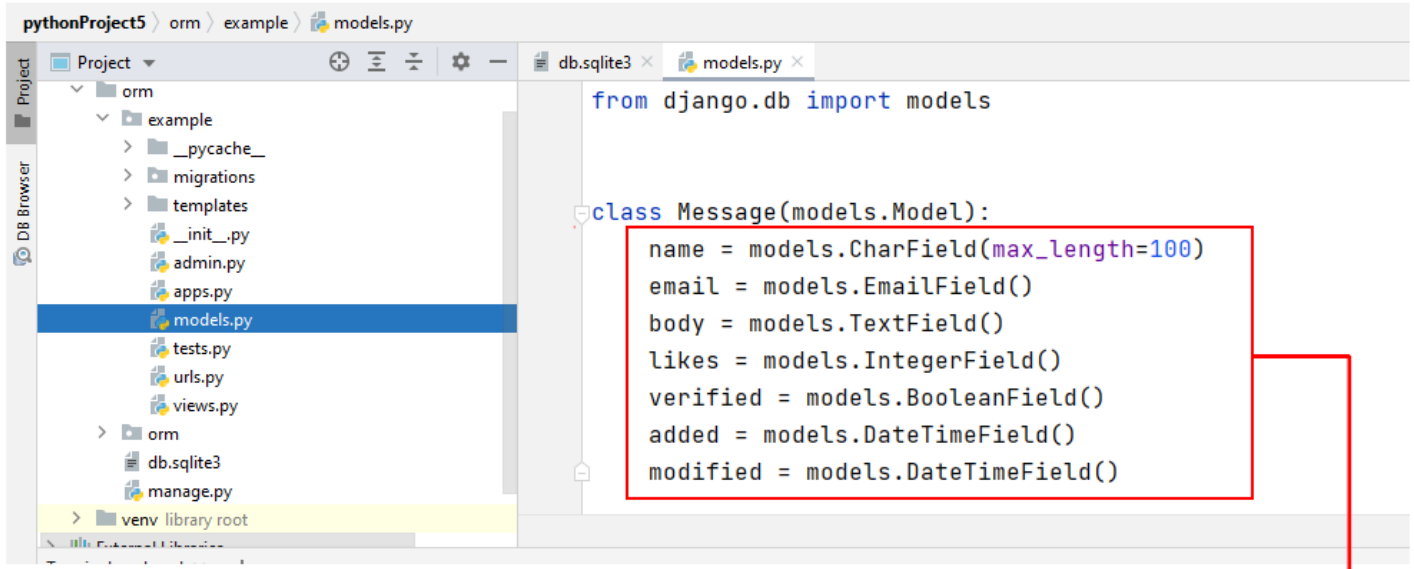
## WPIS Z MIGRACJA

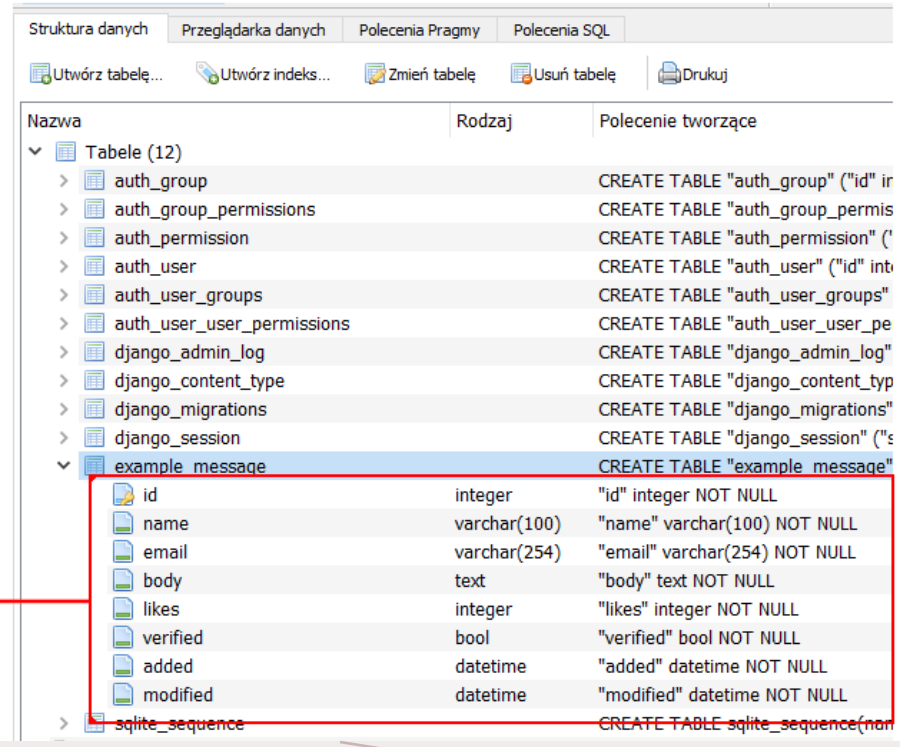



# DOMYSLNA KONWENCJA NAZEWNICZA TABELI

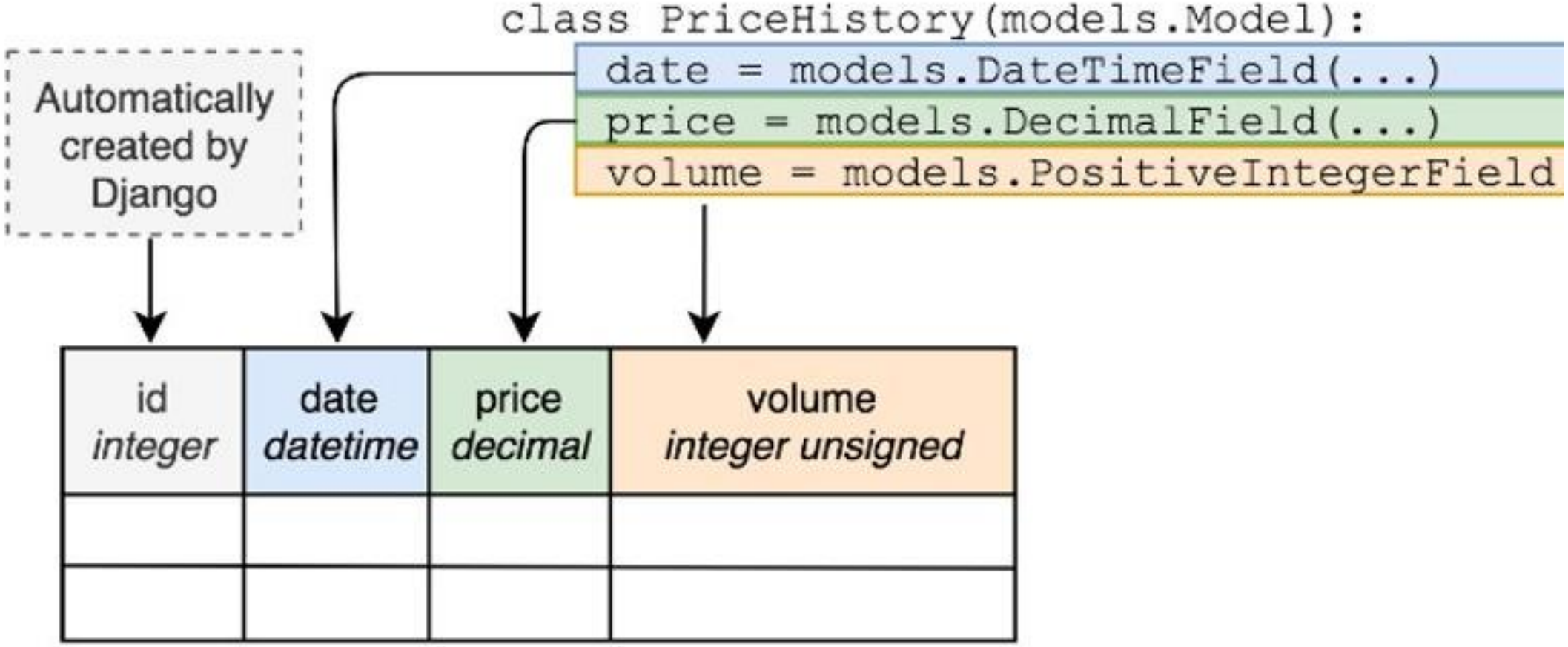
*<NAZWA APLIKACJI>_<NAZWA MODELU>*




# DOMYSLNA KONWENCJA NAZEWNICZA PÓL TABELI

*<NAZWA ATRYBUTU MODELU>*




# DOMYSLNA KONWENCJA NAZEWNICZA PÓL TABELI

*<NAZWA ATRYBUTU MODELU>*

# *INSTRUKCJE DDL*

## *PRZEGLAD WYBRANYCH TYPÓW PÓL*

# DOSTEPNE TYPY PÓL

| | | | | | |
|---|---|---|---|---|---|
| AutoField | **CharField** | **EmailField** | **IntegerField** | PositiveIntegerField | **TextField** |
| BigAutoField | DateField | **FileField** | GenericIPAddressField | PositiveSmallIntegerField | TimeField |
| BigIntegerField | **DateTimeField** | FilePathField | **JSONField** | SlugField | URLField |
| BinaryField | **DecimalField** | FloatField | NullBooleanField | SmallAutoField | UUIDField |
| **BooleanField** | DurationField | **ImageField** | PositiveBigIntegerFIeld | SmallIntegerField | + Relacje |

# BooleanField

Pole do przechowywania wartości logicznej: prawda lub fałsz. Pole jest rzutowane na kolumnę typu bool dla baz sqlite3 i mysql, na kolumnę typu boolean dla bazy postgresql oraz na kolumnę typu NUMBER(1) dla bazy oracle.

# CharField

Pole do przechowywania krótkiej lub średniej długości wartości tekstowej. Posiada jeden obowiązkowy parametr – **max_length** opisujący maksymalną długość przechowywanego napisu. Pole rzutowane jest na kolumnę typu VARCHAR dla baz sqlite3, postgresql i mysql oraz na kolumnę typu NVARCHAR2 dla bazy oracle.

# DateTimeField

Pole do przechowywania obiektu klasy datetime.datetime. W Django istnieją trzy, wbudowane pola do przechowywania informacji o dacie/czasie - DateField, DateTimeField, TimeField. Pola te posiadają dwa wspólne, opcjonalne parametry:

- **auto_now_add** - parametr wykorzystywany do przechowywania daty stworzenia wpisu. Kiedy ustawiony na true w momencie **pierwszego** zapisania instancji modelu w polu wpisywana jest aktualna data/czas.
- **auto_now** – parametr wykorzystywany do przechowywania daty ostatniej modyfikacji wpisu. Kiedy ustawiony na true w polu wpisywana jest automatycznie wartość z aktualną datą/czasem **za każdym razem**, kiedy na danej instancji modelu wywoływana jest metoda save.

Pole rzutowane jest na kolumnę typu datetime dla bazy sqlite3, datetime(6) dla bazy mysql, timestamp with time zone dla bazy postgresql oraz TIMESTAMP dla bazy oracle.

# DecimalField

Pole do przechowywania wartość zmiennoprzecinkowej **z ustaloną precyzją** (instancja klasy Decimal). W odróżnieniu od pola typu FloatField umożliwia wykonywanie operacji matematycznych z określoną precyzją. Posiada dwa **obowiązkowe** parametry:

- **max_digits** – maksymalna liczba cyfr w zapisywanej liczbie. Wartość musi być większa lub równa wartości parametru decimal_places
- **decimal_places** – liczba 'miejsc po przecinku'  w zapisywanej liczbie.

Na przykład, zapis:

Models.DecimalField(…, max_digits=5, decimal_places=2)

pozwala na przechowywanie w polu wartości z zakresu od –999 do 999 z dokładnością do dwóch miejsc po przecinku. Pole rzutowane jest na kolumnę typu decimal dla bazy sqlite3, numeric dla baz mysql i postgresql oraz NUMBER dla bazy oracle.

# EmailField

Pole do przechowywania krótkiej lub średniej długości wartości tekstowej w postacie email. Pole jest rzutowane na kolumnę tabeli identycznie jak pole typu CharField.

# FileField

Pole do przechowywania plików. Ze względów optymalizacyjnych pliki nie są przechowywane w bazie, tylko na serwerze aplikacji lub CDN, a w bazie przechowywany jest wyłącznie adres tych plików. Posiada dwa opcjonalne parametry:

- upload_to – parametr wskazujący na folder, w którym mają być przechowywane pliki oraz sposób nadawania im nazw. Więcej szczegółów można znaleźć tutaj
- storage – obiekt obsługujacy zapisywanie oraz odczytywanie plików. Szczegółowy opis parametru można znaleźć tutaj

Pole jest rzutowane na kolumnę tabeli identycznie jak pole typu CharField.

# ImageField

Pole do przechowywania pliku graficznego. Dziedziczy parametry pola FileField. Dodatkowo sprawdza, czy zapisywany plik jest prawidłowym plikiem graficznym. Pole jest rzutowane na kolumnę tabeli identycznie jak pole typu CharField.

# IntegerField

Pole do przechowywania wartości całkowitoliczbowej z zakresu od -2147483648 do 2147483647. Rzutowane jest na kolumnę typu integer dla bazy sqlite3, mysql, postgres oraz na kolumnę typu NUMBER(11) dla bazy oracle.

# JsonField

Pole do przechowywania wartości w formacie json. Rzutowane jest na kolumnę typu text dla bazy sqlite3, json dla bazy mysql, jsonb dla bazy postgres oraz na kolumnę typu NCLOB dla bazy oracle.

# TextField

Pole do przechowywania długich wartości tekstowych. Rzutowane jest na kolumnę typu text dla baz sqlite3 i postgres, longtext dla bazy mysql oraz na kolumnę typu NCLOB dla bazy oracle.

# *INSTRUKCJE DDL*

## *PRZEGLAD WYBRANYCH PARAMETRÓW PÓL*

# *DOSTEPNE PARAMETRY PÓL*

Istnieje 17 parametrów dostępnych dla **wszystkich** typów pól. Poszczególne typy mogą posiadać dodatkowe, własne parametry (np. parametr max_lenght pola CharField).

| | | | | | |
|---|---|---|---|---|---|
| **null** | db_column | **default** | help_text | unique_for_date | **verbose_name** |
| **blank** | db_index | editable | primary_key | unique_for_month | validators |
| choices | db_tablespace | error_messages | **unique** | unique_for_year | |

# *NULL*

Parametr o wartości binarnej. Ustawiony na True umożliwia **zapisanie** pustej wartości (tj, null) **w bazie**, w odpowiadającej temu prametrowi kolumnie. Domyślnie wszystkie pola mają ustawioną wartość parametru null na False.

# *BLANK*

Parametr o wartości binarnej. Ustawiony na True umożliwia **wprowadzanie** pustej wartości **w formularzu modelu** (patrz prezentacja – widoki). Domyślnie wszystkie pola mają ustawioną wartość parametru blank na False.

# *DEFAULT*

Parametr wskazujący domyślną wartość pola. Wartość domyślna jest przypisywana polu w trakcie tworzenia obiektu modelu, kiedy nie zostanie dostarczona wartość dla danego pola.

# *UNIQUE*

Parametr o wartości biarnej. Ustawiony na True wymusza unikalność wartości w danej kolumnie.

# *VERBOSE_NAME*

Parametr opisujący nazwę kolumny. Jeżeli parametr nie jest ustaniowny ORM Django przypisuje kolumnie nazwę identyczną z nazwą wprowadzonego w modelu atrybutu.

# *INSTRUKCJE DDL*

## *RELACJE*

# *DOSTEPNE TYPY RELACJI*

| |
|---|
| OneToOneField |
| ForeignKey |
| ManyToManyField |

# *OBOWIAZKOWE PARAMETRY PÓL RELACJI*

Wszystkie pola reprezenentujące relacje posiadają dwa obowiązkowe parametry:

- **nazwa modelu**, do którego odwołuje się relacja
- **on_delete** – parametr opisujący akcje wykonywaną na wpisie, kiedy wpis powiązany (tj. ten do którego się odwołuje) zostanie usunięty (nie dotyczy relacji ManyToManyField). Dostępne opcje to: CASCADE, PROTECT, RESTRICT, SET_NULL, SET_DEFAULT, SET(), DO_NOTHING

# OneToOneField

Pole reprezentujące relacje jeden do jednego (ang. one-to-one). Często wykorzystywane jako klucz obcy do modelu rozszerzającego podstawowy model o dodatkowe informacje.

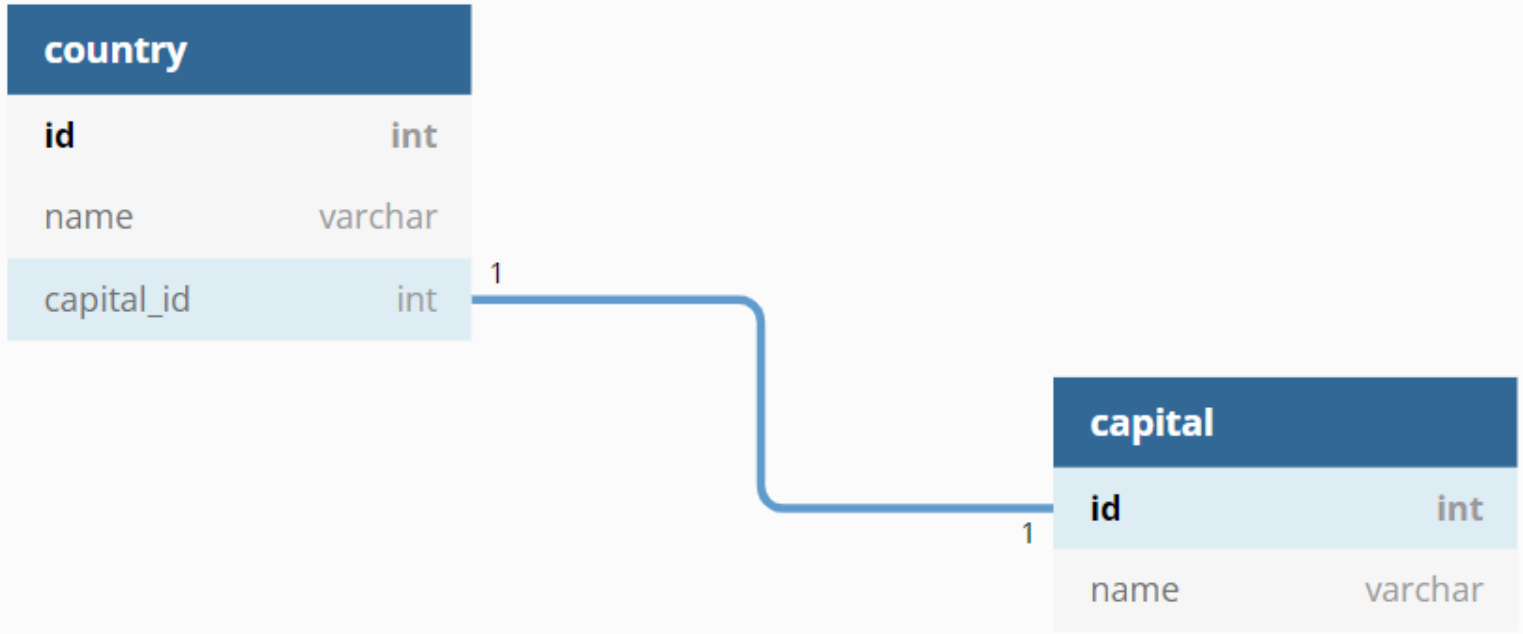
Pole OneToOneField posiada 2 dodatkowe parametry opcjonalne: to_field, parent_link. Więcej informacji o polu można znaleźć tutaj

https://dbdiagram.io/d/61a14d248c901501c0d4a0d1

Zaloguj

Untitled Diagram Saved | Save | Share | History | Export | Import | holistics.io | Upgrade | Help

```
//// -- RELATIONS
//// -- one-to-one

Table country as Co {
  id int [pk, increment]
  name varchar
  capital_id int
}

Table capital as Ca {
  id int [pk, increment]
  name varchar
 }

Ref: Co.capital_id - capital.id
```

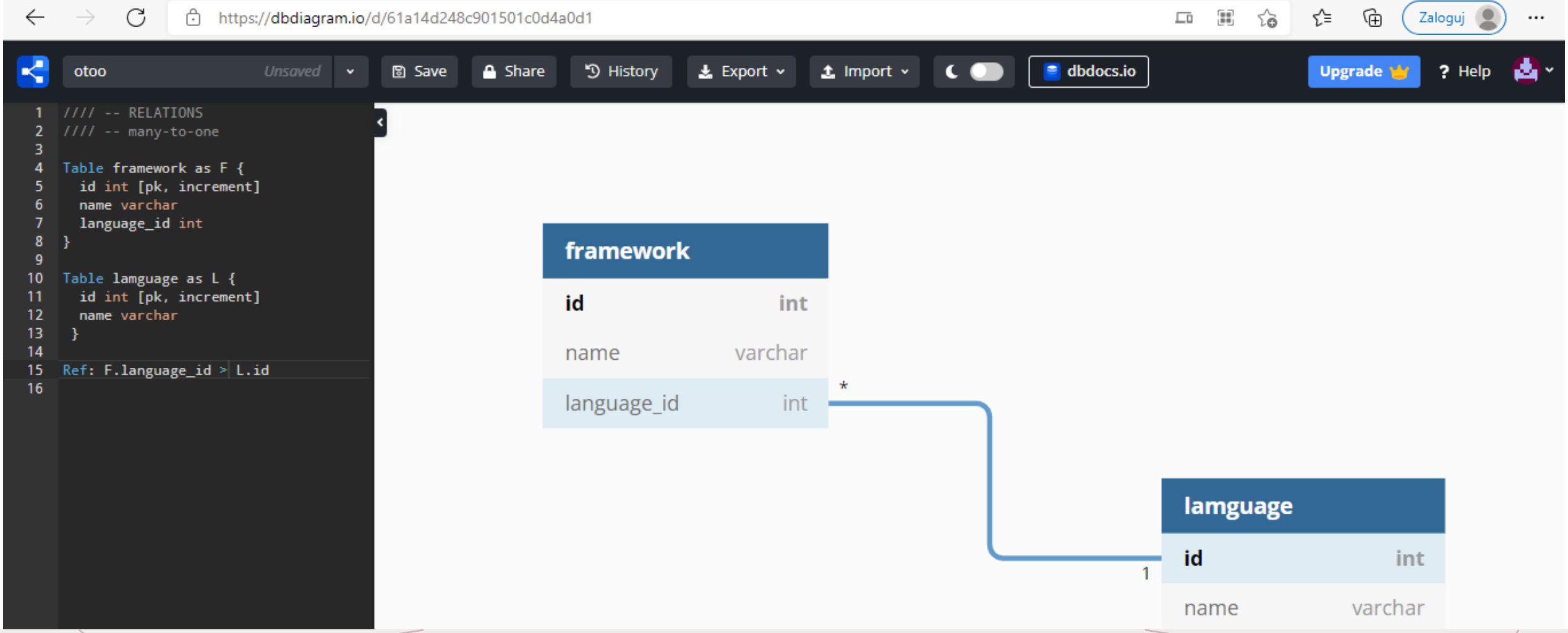



# OneToOneField

# ForeignKey

Pole reprezentujące relacje wiele do jednego (ang. many-to-one)

Pole ForeignKey posiada 1 dodatkowy, parametr opcjonalny  - to_field. Więcej informacji o polu można znaleźć tutaj

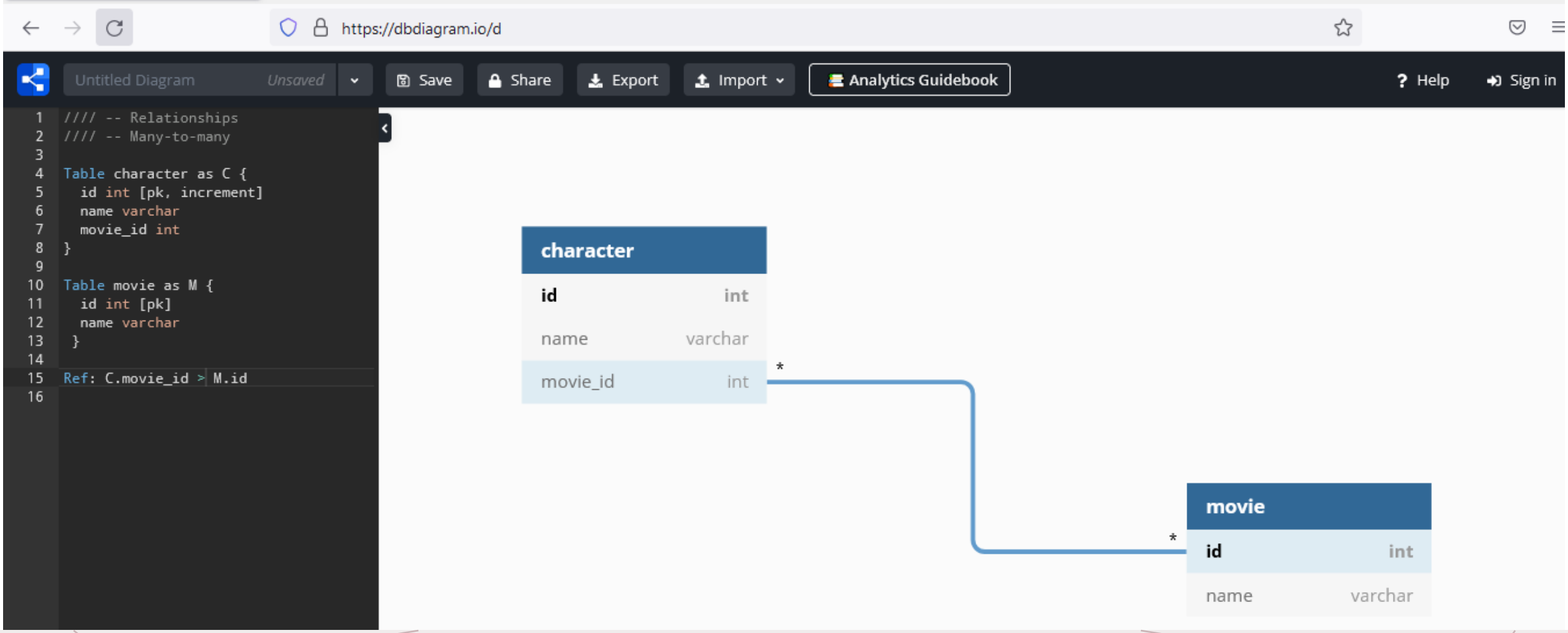

https://dbdiagram.io/d/61a14d248c901501c0d4a0d1
Zaloguj
otoo
Unsaved
Save
Share
History
Export
Import
dbdocs.io
Upgrade
Help
//// -- RELATIONS
//// -- many-to-one
Table framework as F {
id int [pk, increment]
name varchar
language_id int
}
Table lamguage as L {
id int [pk, increment]
name varchar
}
Ref: F.language_id > L.id
framework
id int
name varchar
language_id int
*
1
lamguage
id int
name varchar

# ForeignKey

# ManyToManyField

Pole reprezentujące relacje wiele do wielu (ang. many-to-many).

Pole ManyToManyField posiada 4 dodatkowe parametry opcjonalne: symmetrical, through, through_fields, db_table. Więcej informacji o polu można znaleźć tutaj

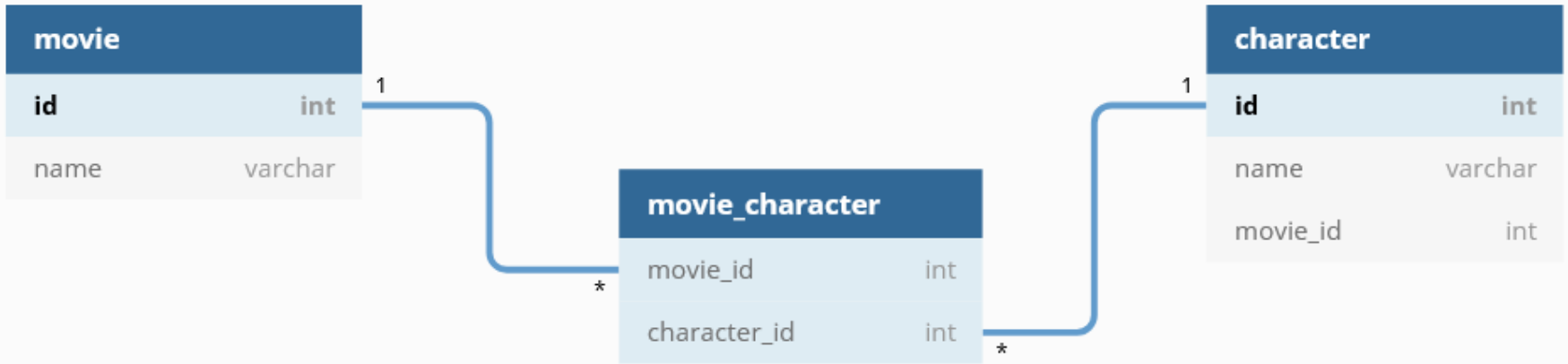



# ManyToManyField

https://dbdiagram.io/d

Untitled Diagram *Unsaved* | Save | Share | Export | Import | Analytics Guidebook | Help | Sign in

```
//// -- Relationships
//// -- Many-to-many

Table character as C {
  id int [pk, increment]
  name varchar
  movie_id int
}

Table movie as M {
  id int [pk]
  name varchar
 }

 Table movie_character as MC {
   movie_id int
   character_id int
 }
Ref: MC.movie_id > M.id
Ref: MC.character_id > C.id
```




# ManyToManyField

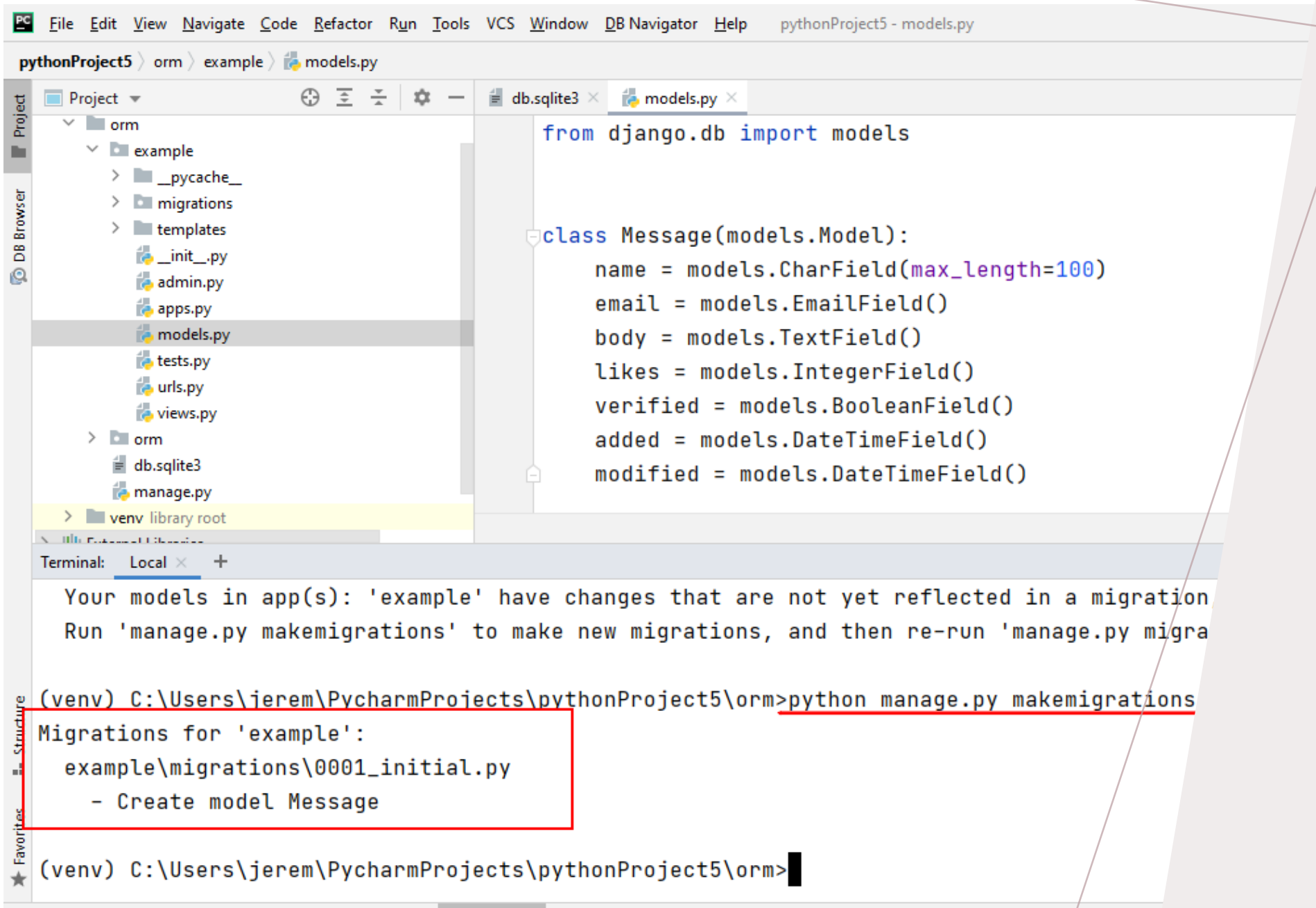
# *OPCJONALNE PARAMETRY PÓL RELACJI*

Istnieje 5 parametrów dostępnych dla **wszystkich** typów relacji. Poszczególne relacje mogą posiadać dodatkowe, własne parametry (np. parametr through relacji ManyToMany).

| | |
|---|---|
| limit_choices_to | db_constraint |
| related_name | swappable |
| related_query_name | |

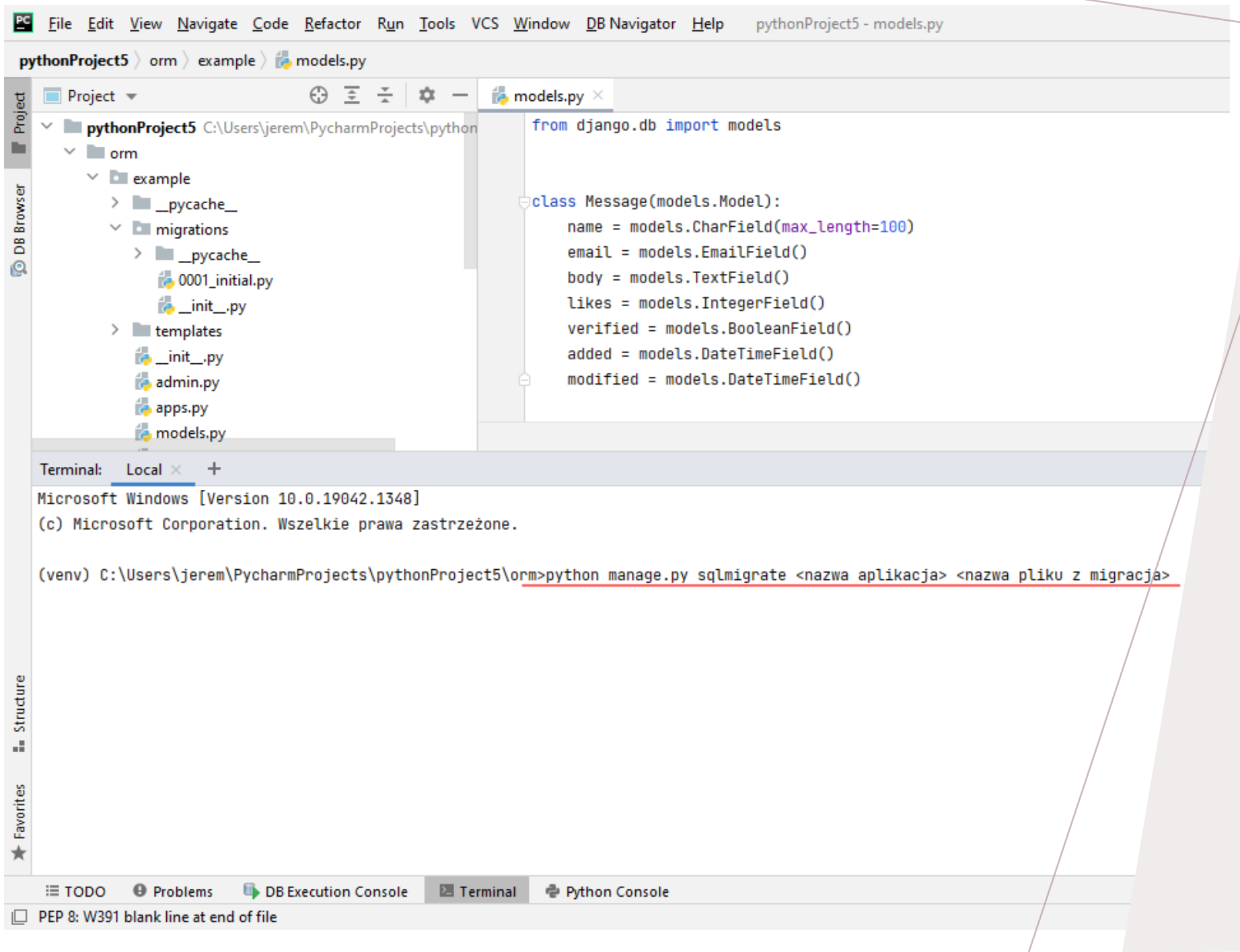
# *INSTRUKCJE DDL*

## *CREATE TABLE*



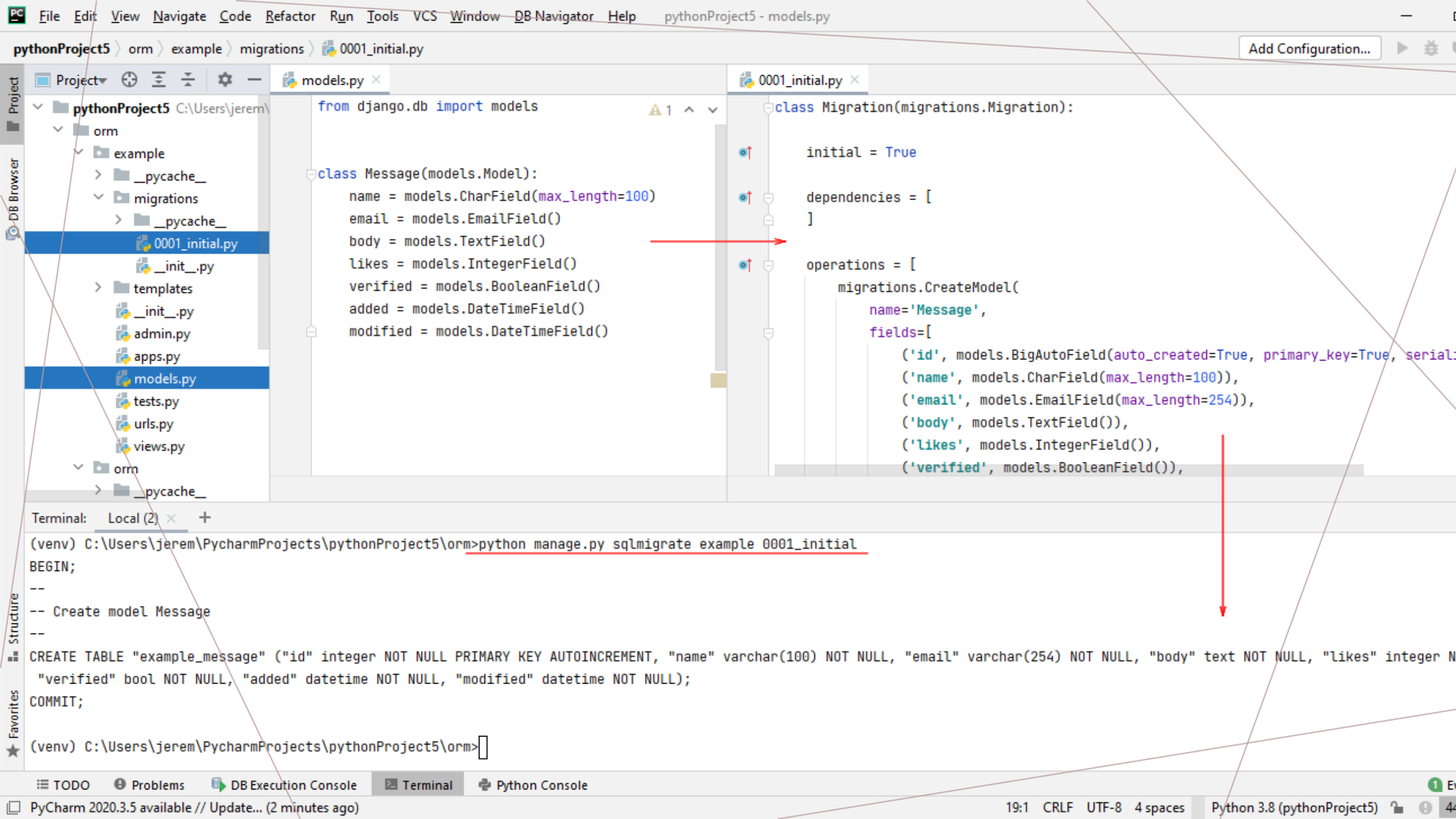

# MODELE

## MAKEMIGRATIONS



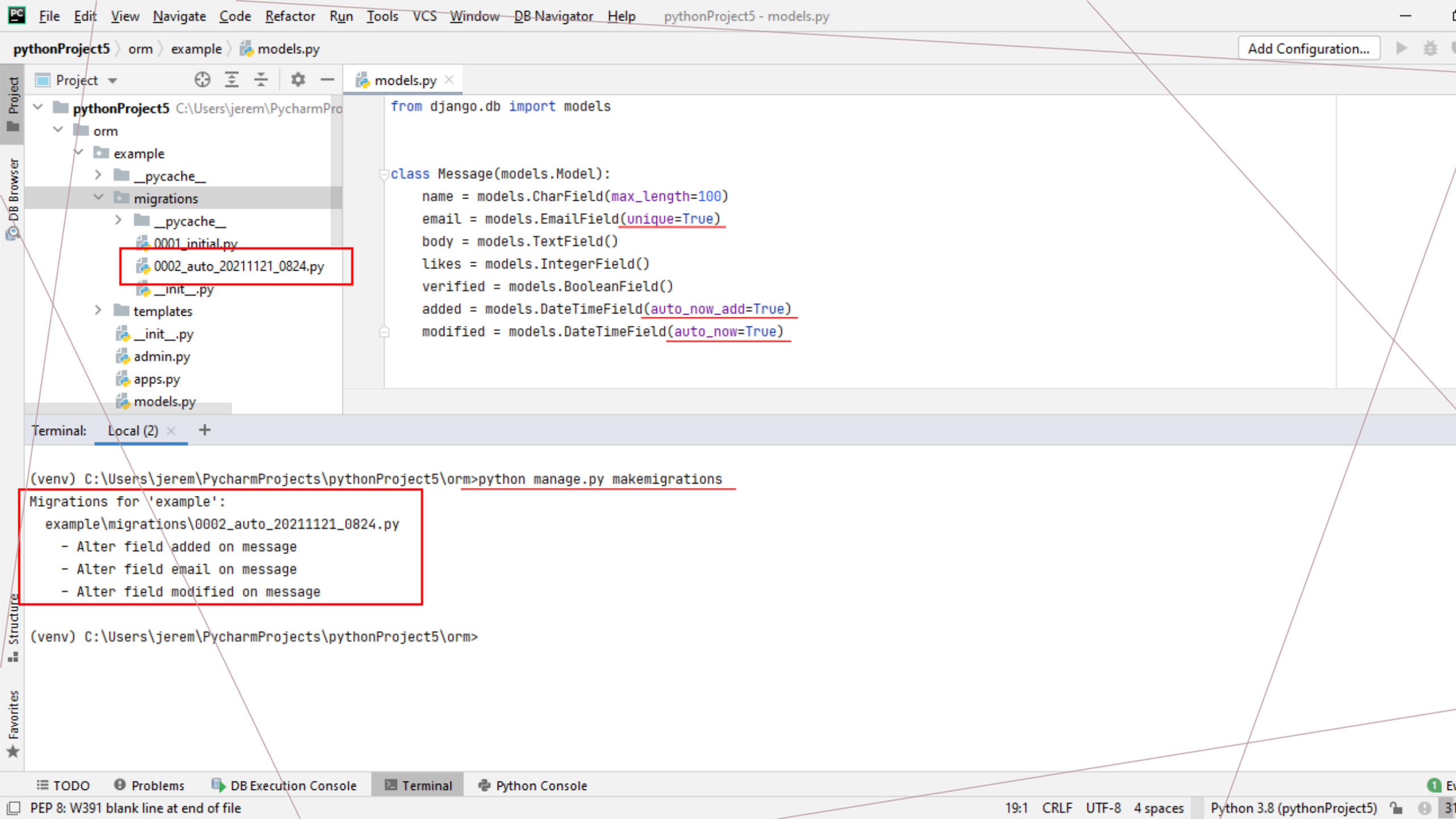

# MIGRACJE

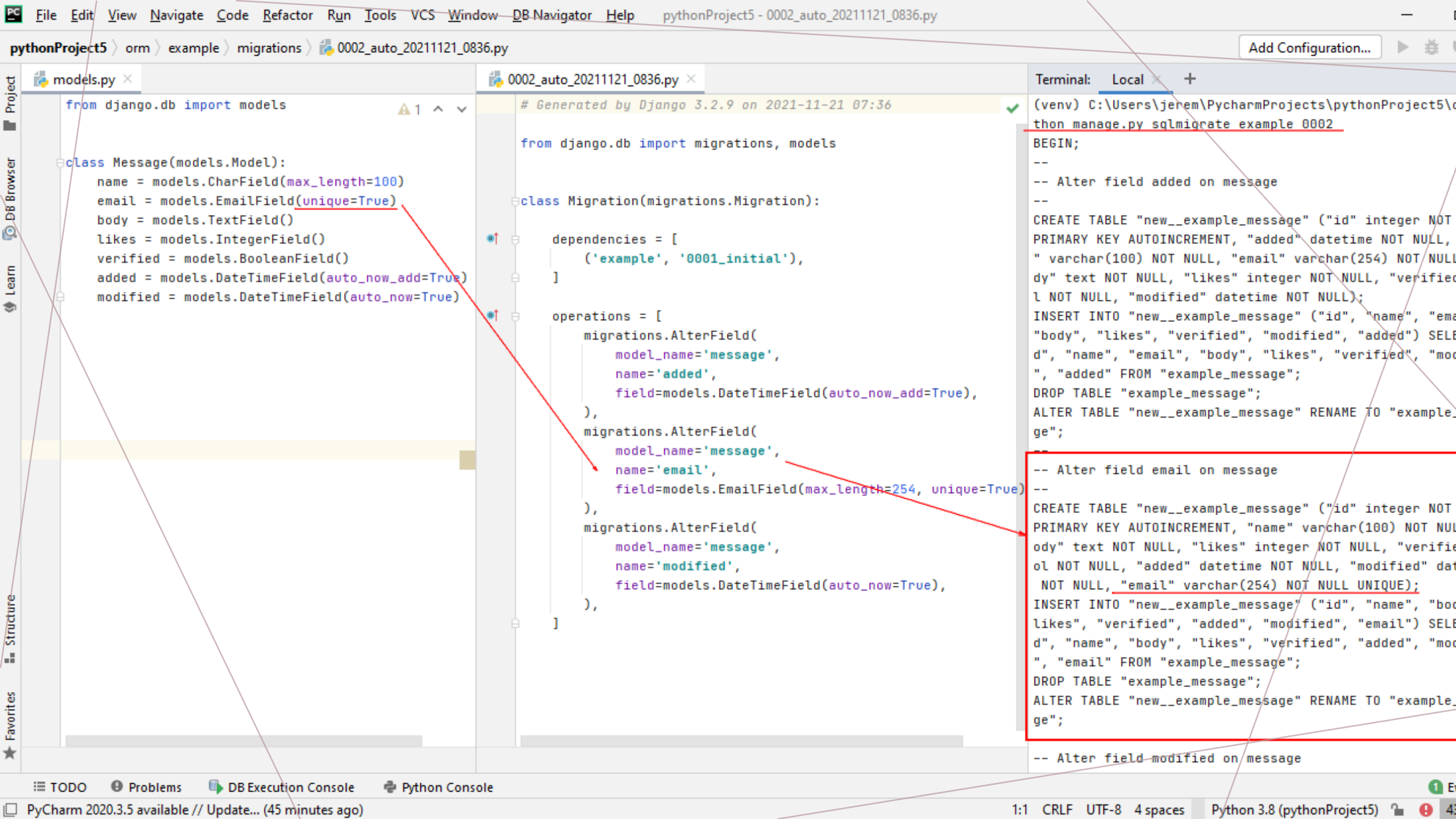
## SQLMIGRATE

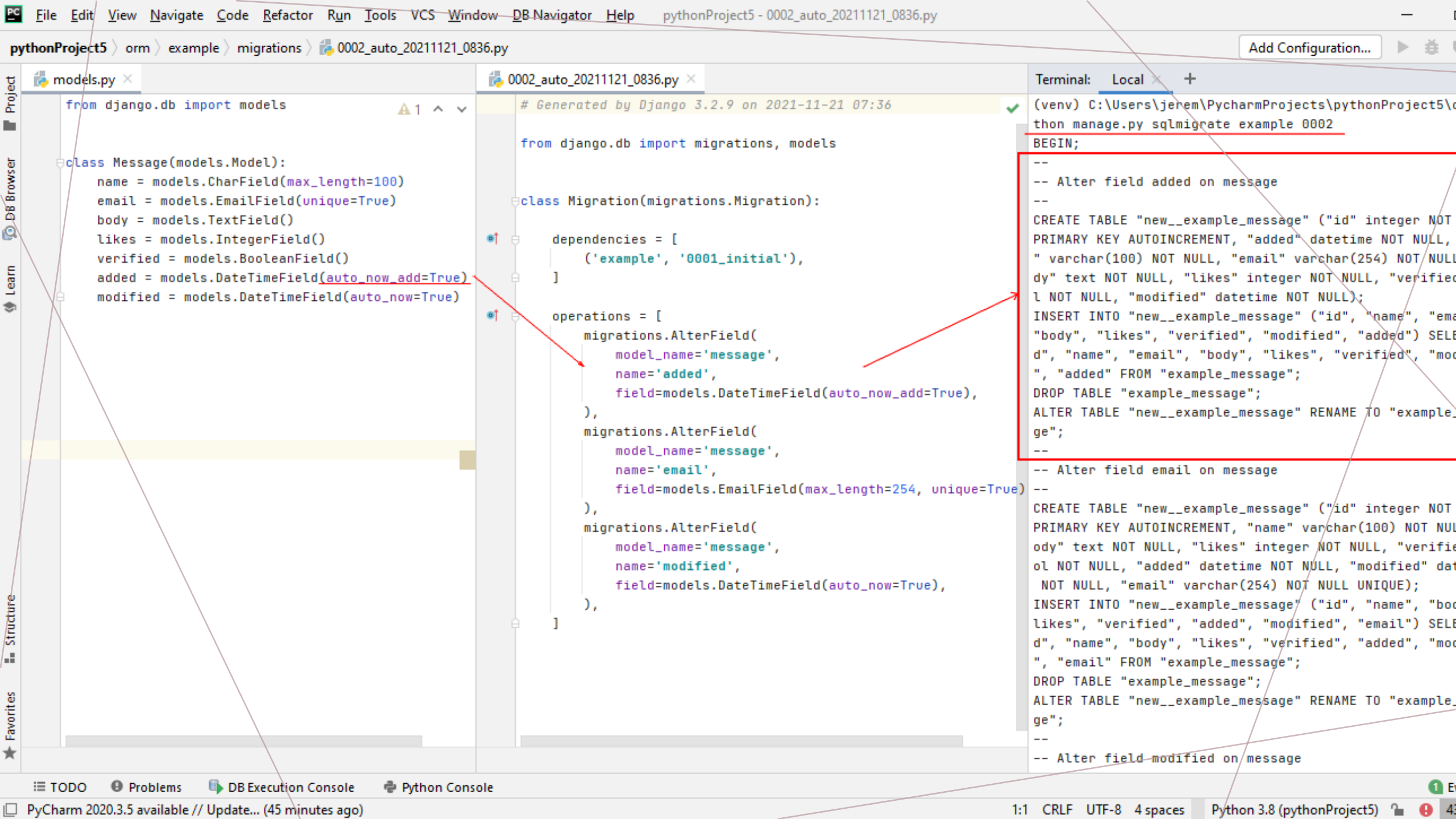

File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help
pythonProject5 - models.py
pythonProject5 〉 orm 〉 example 〉 migrations 〉 0001_initial.py
Add Configuration...
Project
pythonProject5 C:\Users\jerem\
orm
example
__pycache__
migrations
__pycache__
0001_initial.py
__init__.py
templates
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
orm
__pycache__
models.py
from django.db import models
class Message(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField()
body = models.TextField()
likes = models.IntegerField()
verified = models.BooleanField()
added = models.DateTimeField()
modified = models.DateTimeField()
0001_initial.py
class Migration(migrations.Migration):
initial = True
dependencies = [
]
operations = [
migrations.CreateModel(
name='Message',
fields=[
('id', models.BigAutoField(auto_created=True, primary_key=True, serial
('name', models.CharField(max_length=100)),
('email', models.EmailField(max_length=254)),
('body', models.TextField()),
('likes', models.IntegerField()),
('verified', models.BooleanField()),
Terminal: Local (2)
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\orm>python manage.py sqlmigrate example 0001_initial
BEGIN;
--
-- Create model Message
--
CREATE TABLE "example_message" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(100) NOT NULL, "email" varchar(254) NOT NULL, "body" text NOT NULL, "likes" integer N
"verified" bool NOT NULL, "added" datetime NOT NULL, "modified" datetime NOT NULL);
COMMIT;
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\orm>
TODO
Problems
DB Execution Console
Terminal
Python Console
PyCharm 2020.3.5 available // Update... (2 minutes ago)
19:1 CRLF UTF-8 4 spaces
Python 3.8 (pythonProject5)

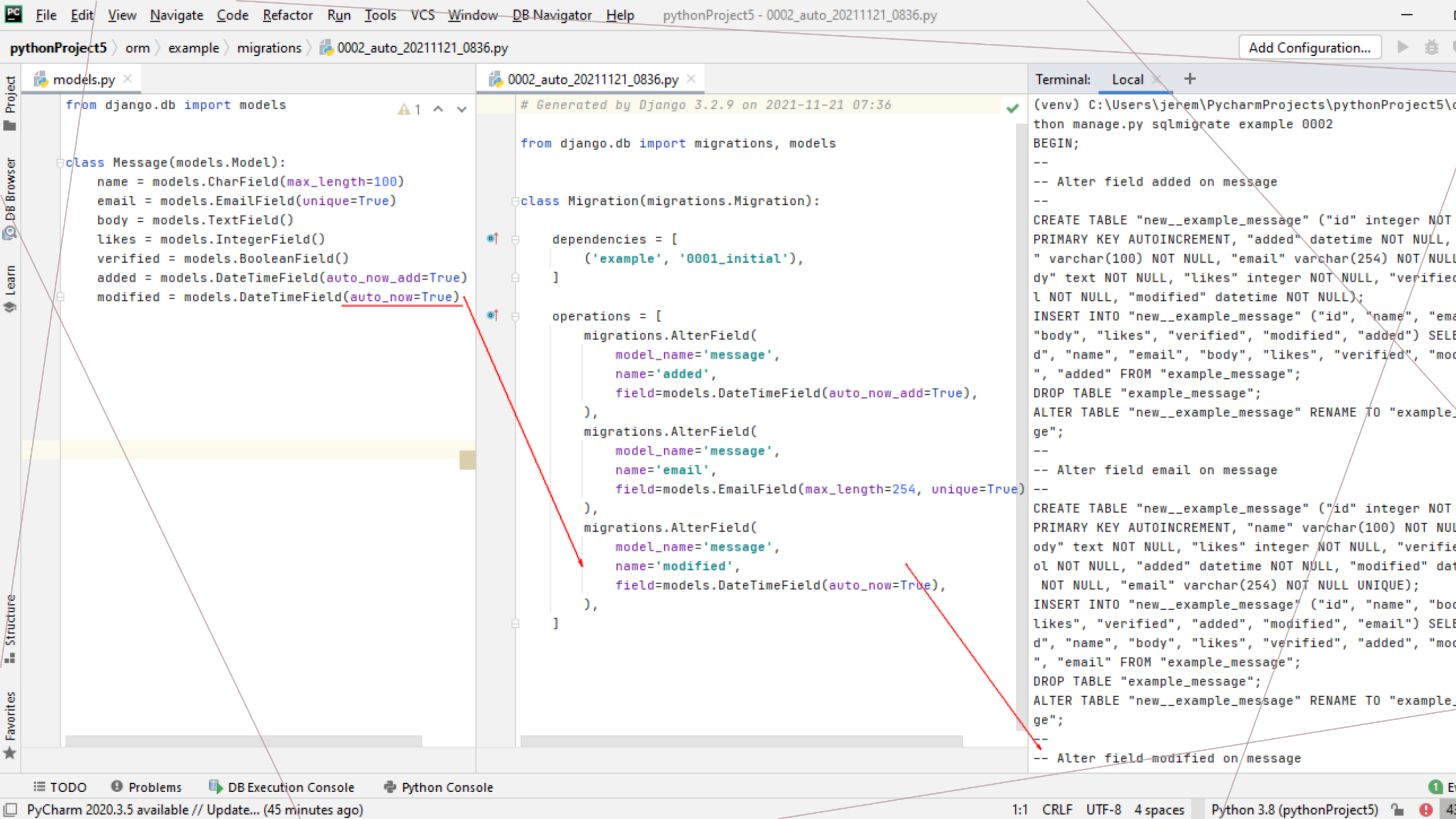
# *INSTRUKCJE DDL*

## *ALTER TABLE*


File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help
pythonProject5 - models.py
pythonProject5 〉 orm 〉 example 〉 models.py
Add Configuration...
Project
pythonProject5 C:\Users\jerem\PycharmPro
orm
example
__pycache__
migrations
__pycache__
0001_initial.py
0002_auto_20211121_0824.py
__init__.py
templates
__init__.py
admin.py
apps.py
models.py
models.py
from django.db import models
class Message(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField(unique=True)
body = models.TextField()
likes = models.IntegerField()
verified = models.BooleanField()
added = models.DateTimeField(auto_now_add=True)
modified = models.DateTimeField(auto_now=True)
Terminal: Local (2)
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\orm>python manage.py makemigrations
Migrations for 'example':
example\migrations\0002_auto_20211121_0824.py
- Alter field added on message
- Alter field email on message
- Alter field modified on message
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\orm>
TODO
Problems
DB Execution Console
Terminal
Python Console
PEP 8: W391 blank line at end of file
19:1 CRLF UTF-8 4 spaces
Python 3.8 (pythonProject5)


File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help
pythonProject5 - 0002_auto_20211121_0836.py
pythonProject5 orm example migrations 0002_auto_20211121_0836.py
Add Configuration...
models.py
from django.db import models
class Message(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField(unique=True)
body = models.TextField()
likes = models.IntegerField()
verified = models.BooleanField()
added = models.DateTimeField(auto_now_add=True)
modified = models.DateTimeField(auto_now=True)
0002_auto_20211121_0836.py
# Generated by Django 3.2.9 on 2021-11-21 07:36
from django.db import migrations, models
class Migration(migrations.Migration):
dependencies = [
('example', '0001_initial'),
]
operations = [
migrations.AlterField(
model_name='message',
name='added',
field=models.DateTimeField(auto_now_add=True),
),
migrations.AlterField(
model_name='message',
name='email',
field=models.EmailField(max_length=254, unique=True)
),
migrations.AlterField(
model_name='message',
name='modified',
field=models.DateTimeField(auto_now=True),
),
]
Terminal: Local
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\o
thon manage.py sqlmigrate example 0002
BEGIN;
--
-- Alter field added on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "added" datetime NOT NULL,
" varchar(100) NOT NULL, "email" varchar(254) NOT NULL
dy" text NOT NULL, "likes" integer NOT NULL, "verified
l NOT NULL, "modified" datetime NOT NULL);
INSERT INTO "new__example_message" ("id", "name", "ema
"body", "likes", "verified", "modified", "added") SELE
d", "name", "email", "body", "likes", "verified", "mod
", "added" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
-- Alter field email on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "name" varchar(100) NOT NUL
ody" text NOT NULL, "likes" integer NOT NULL, "verifie
ol NOT NULL, "added" datetime NOT NULL, "modified" dat
NOT NULL, "email" varchar(254) NOT NULL UNIQUE);
INSERT INTO "new__example_message" ("id", "name", "boo
likes", "verified", "added", "modified", "email") SELE
d", "name", "body", "likes", "verified", "added", "mod
", "email" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
-- Alter field modified on message
TODO Problems DB Execution Console Python Console
PyCharm 2020.3.5 available // Update... (45 minutes ago)
1:1 CRLF UTF-8 4 spaces Python 3.8 (pythonProject5)


File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help
pythonProject5 - 0002_auto_20211121_0836.py
pythonProject5 orm example migrations 0002_auto_20211121_0836.py
Add Configuration...
models.py
from django.db import models
class Message(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField(unique=True)
body = models.TextField()
likes = models.IntegerField()
verified = models.BooleanField()
added = models.DateTimeField(auto_now_add=True)
modified = models.DateTimeField(auto_now=True)
0002_auto_20211121_0836.py
# Generated by Django 3.2.9 on 2021-11-21 07:36
from django.db import migrations, models
class Migration(migrations.Migration):
dependencies = [
('example', '0001_initial'),
]
operations = [
migrations.AlterField(
model_name='message',
name='added',
field=models.DateTimeField(auto_now_add=True),
),
migrations.AlterField(
model_name='message',
name='email',
field=models.EmailField(max_length=254, unique=True)
),
migrations.AlterField(
model_name='message',
name='modified',
field=models.DateTimeField(auto_now=True),
),
]
Terminal: Local
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\o
thon manage.py sqlmigrate example 0002
BEGIN;
--
-- Alter field added on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "added" datetime NOT NULL,
" varchar(100) NOT NULL, "email" varchar(254) NOT NULL
dy" text NOT NULL, "likes" integer NOT NULL, "verified
l NOT NULL, "modified" datetime NOT NULL);
INSERT INTO "new__example_message" ("id", "name", "ema
"body", "likes", "verified", "modified", "added") SELE
d", "name", "email", "body", "likes", "verified", "mod
", "added" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
--
-- Alter field email on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "name" varchar(100) NOT NUL
ody" text NOT NULL, "likes" integer NOT NULL, "verifie
ol NOT NULL, "added" datetime NOT NULL, "modified" dat
NOT NULL, "email" varchar(254) NOT NULL UNIQUE);
INSERT INTO "new__example_message" ("id", "name", "boo
likes", "verified", "added", "modified", "email") SELE
d", "name", "body", "likes", "verified", "added", "mod
", "email" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
--
-- Alter field modified on message
Project
DB Browser
Learn
Structure
Favorites
TODO
Problems
DB Execution Console
Python Console
PyCharm 2020.3.5 available // Update... (45 minutes ago)
1:1 CRLF UTF-8 4 spaces
Python 3.8 (pythonProject5)


File Edit View Navigate Code Refactor Run Tools VCS Window DB Navigator Help
pythonProject5 - 0002_auto_20211121_0836.py
pythonProject5 orm example migrations 0002_auto_20211121_0836.py
Add Configuration...
models.py
from django.db import models
class Message(models.Model):
name = models.CharField(max_length=100)
email = models.EmailField(unique=True)
body = models.TextField()
likes = models.IntegerField()
verified = models.BooleanField()
added = models.DateTimeField(auto_now_add=True)
modified = models.DateTimeField(auto_now=True)
0002_auto_20211121_0836.py
# Generated by Django 3.2.9 on 2021-11-21 07:36
from django.db import migrations, models
class Migration(migrations.Migration):
dependencies = [
('example', '0001_initial'),
]
operations = [
migrations.AlterField(
model_name='message',
name='added',
field=models.DateTimeField(auto_now_add=True),
),
migrations.AlterField(
model_name='message',
name='email',
field=models.EmailField(max_length=254, unique=True)
),
migrations.AlterField(
model_name='message',
name='modified',
field=models.DateTimeField(auto_now=True),
),
]
Terminal: Local
(venv) C:\Users\jerem\PycharmProjects\pythonProject5\o
thon manage.py sqlmigrate example 0002
BEGIN;
--
-- Alter field added on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "added" datetime NOT NULL,
" varchar(100) NOT NULL, "email" varchar(254) NOT NULL
dy" text NOT NULL, "likes" integer NOT NULL, "verified
l NOT NULL, "modified" datetime NOT NULL);
INSERT INTO "new__example_message" ("id", "name", "ema
"body", "likes", "verified", "modified", "added") SELE
d", "name", "email", "body", "likes", "verified", "mod
", "added" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
--
-- Alter field email on message
--
CREATE TABLE "new__example_message" ("id" integer NOT
PRIMARY KEY AUTOINCREMENT, "name" varchar(100) NOT NUL
ody" text NOT NULL, "likes" integer NOT NULL, "verifie
ol NOT NULL, "added" datetime NOT NULL, "modified" dat
NOT NULL, "email" varchar(254) NOT NULL UNIQUE);
INSERT INTO "new__example_message" ("id", "name", "boo
likes", "verified", "added", "modified", "email") SELE
d", "name", "body", "likes", "verified", "added", "mod
", "email" FROM "example_message";
DROP TABLE "example_message";
ALTER TABLE "new__example_message" RENAME TO "example_
ge";
--
-- Alter field modified on message
Project
DB Browser
Learn
Structure
Favorites
TODO
Problems
DB Execution Console
Python Console
PyCharm 2020.3.5 available // Update... (45 minutes ago)
1:1 CRLF UTF-8 4 spaces
Python 3.8 (pythonProject5)

# *INSTRUKCJE DDL*

## *DROP TABLE*

# *INSTRUKCJE DDL*

## *KLASA META*

# *CLASS META*

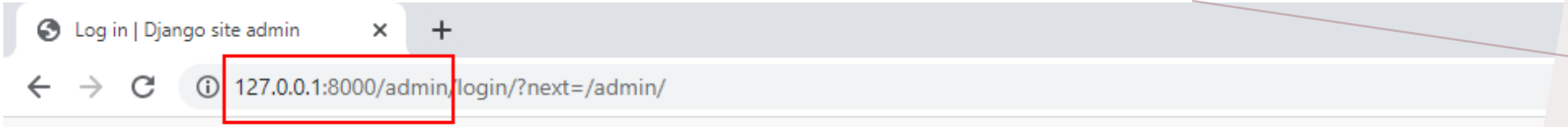
Każdy model posiada klasę Meta. Jest to klasa zagnieżdżona wewnątrz klasy modelu pełniąca rolę pojemnika przechowującego metainformacje o modelu. Informacje przechowywane są w atrybutach klasy Meta. Klasa Meta posiada obecnie 25 wbudowanych atrybutów.

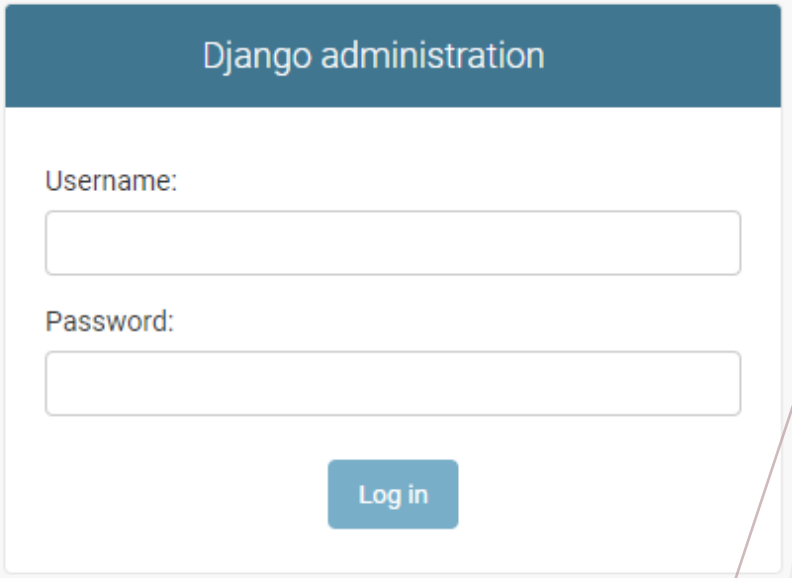
# *DOSTEPNE ATRYBUTY KLASY META*

| | | | | |
|---|---|---|---|---|
| abstract | default_manager_name | **ordering** | required_db_vendor | constraints |
| app_label | default_related_name | **permissions** | select_on_save | **verbose_name** |
| base_manager_name | get_latest_by | default_permissions | **indexes** | verbose_name_plural |
| **db_table** | **managed** | proxy | unique_together | *label* |
| db_tablespace | order_with_respect_to | required_db_feature | index_together | *label_lower* |

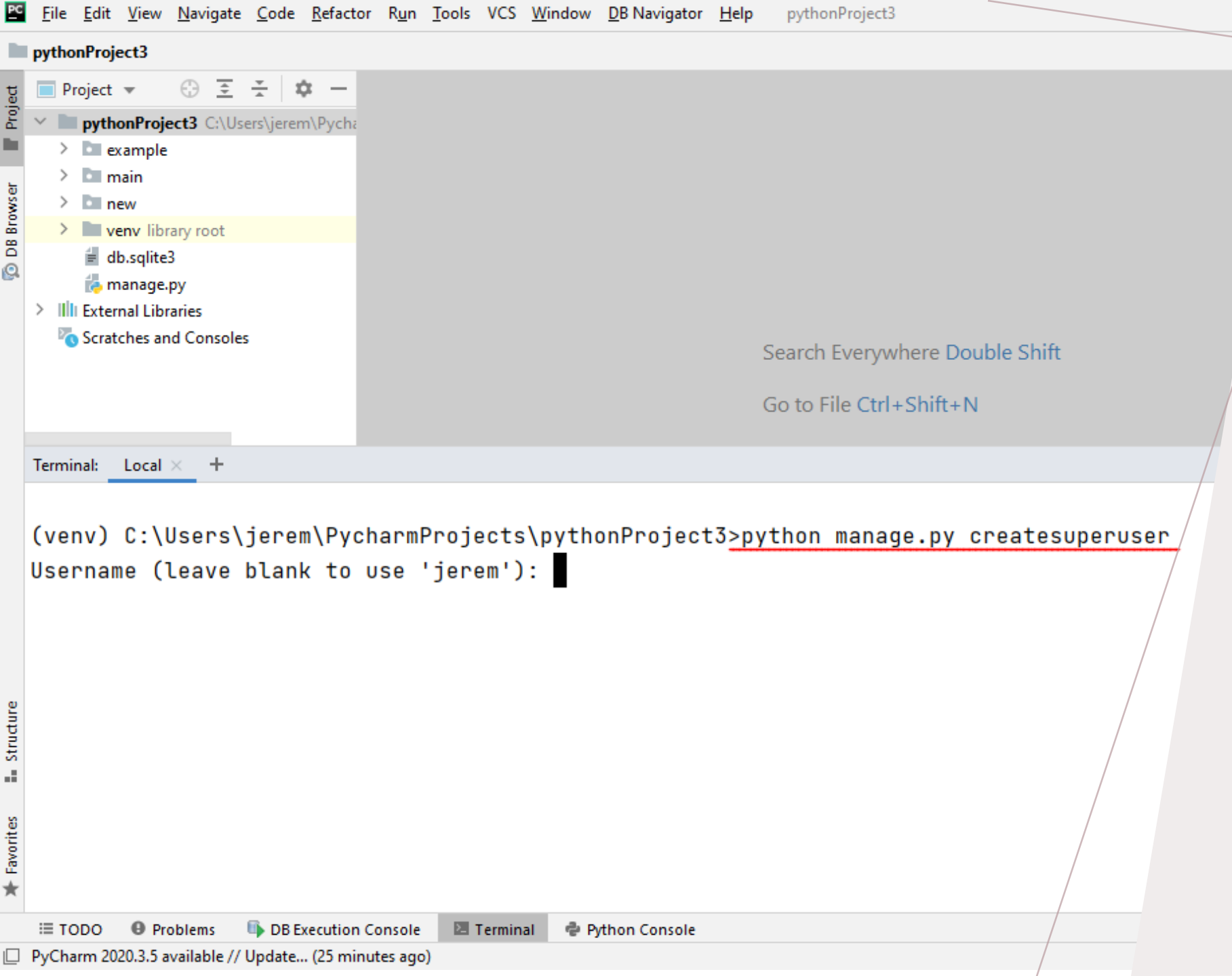
# *DJANGO ORM*

## *PANEL ADMINISTRATORA*



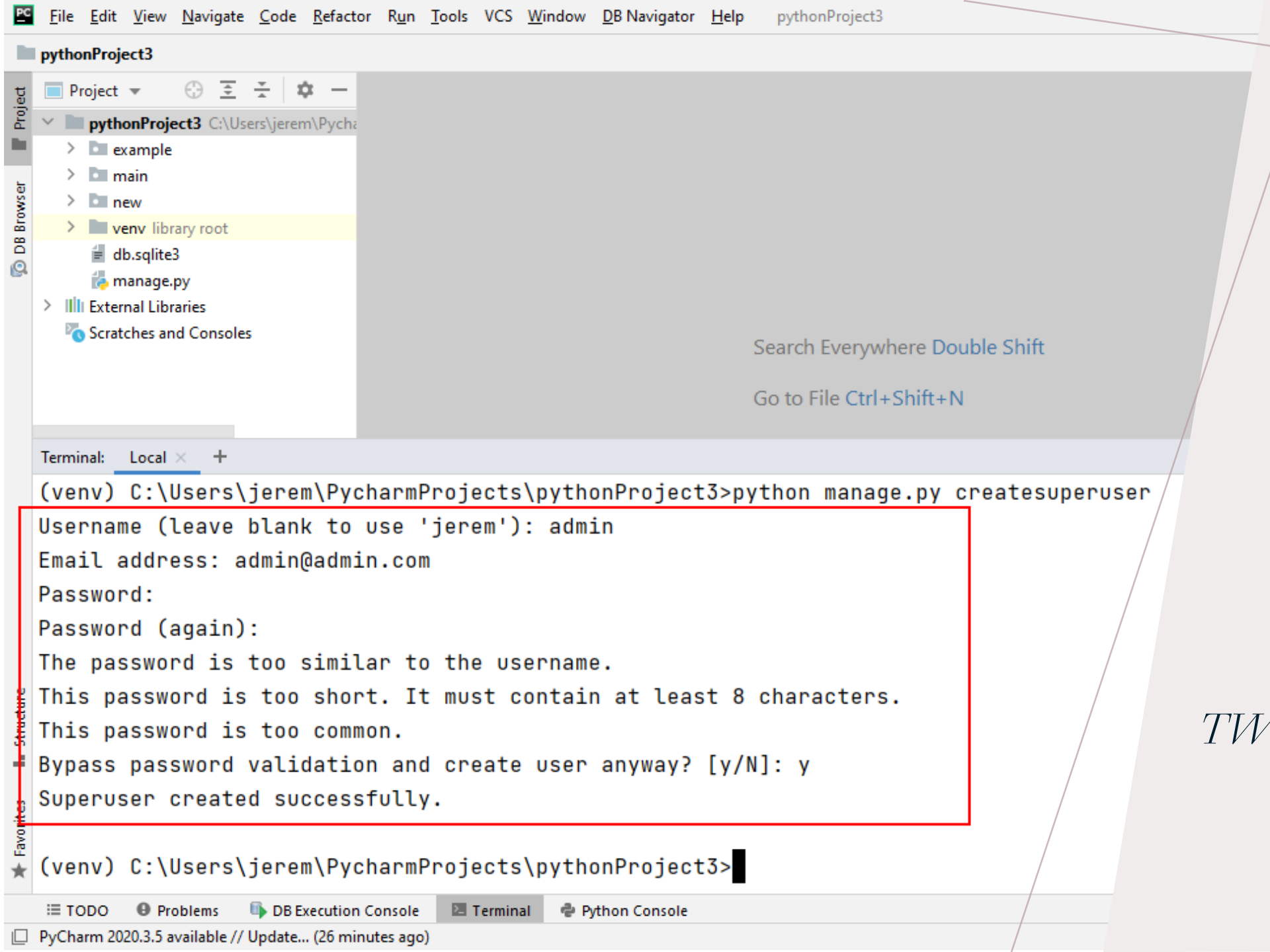




# *LOGOWANIE DO PANELU ADMINISTRATORA*



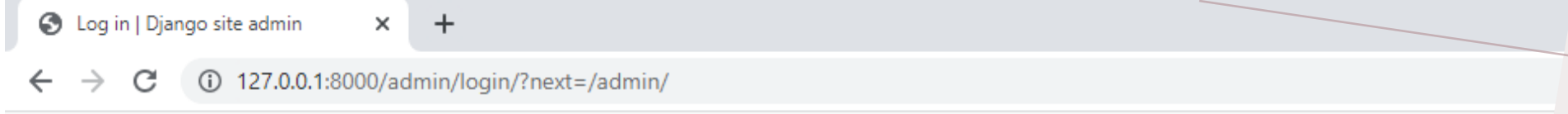

# TWORZENIE SUPER UZYTKOWNIKA

## CREATESUPERUSER




# TWORZENIE SUPER UZYTKOWNIKA

## CREATESUPERUSER




Django administration

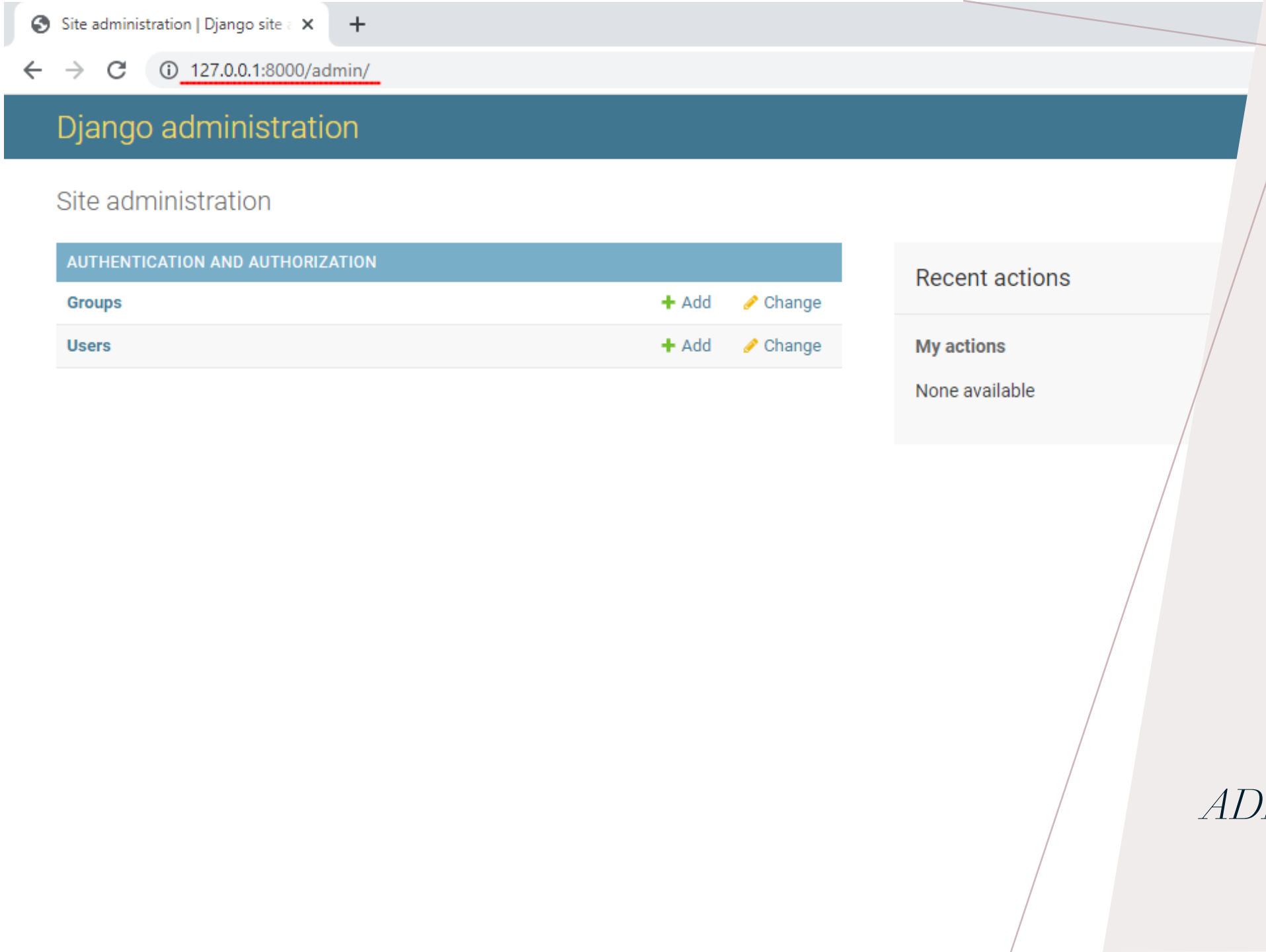
Username:

admin

Password:

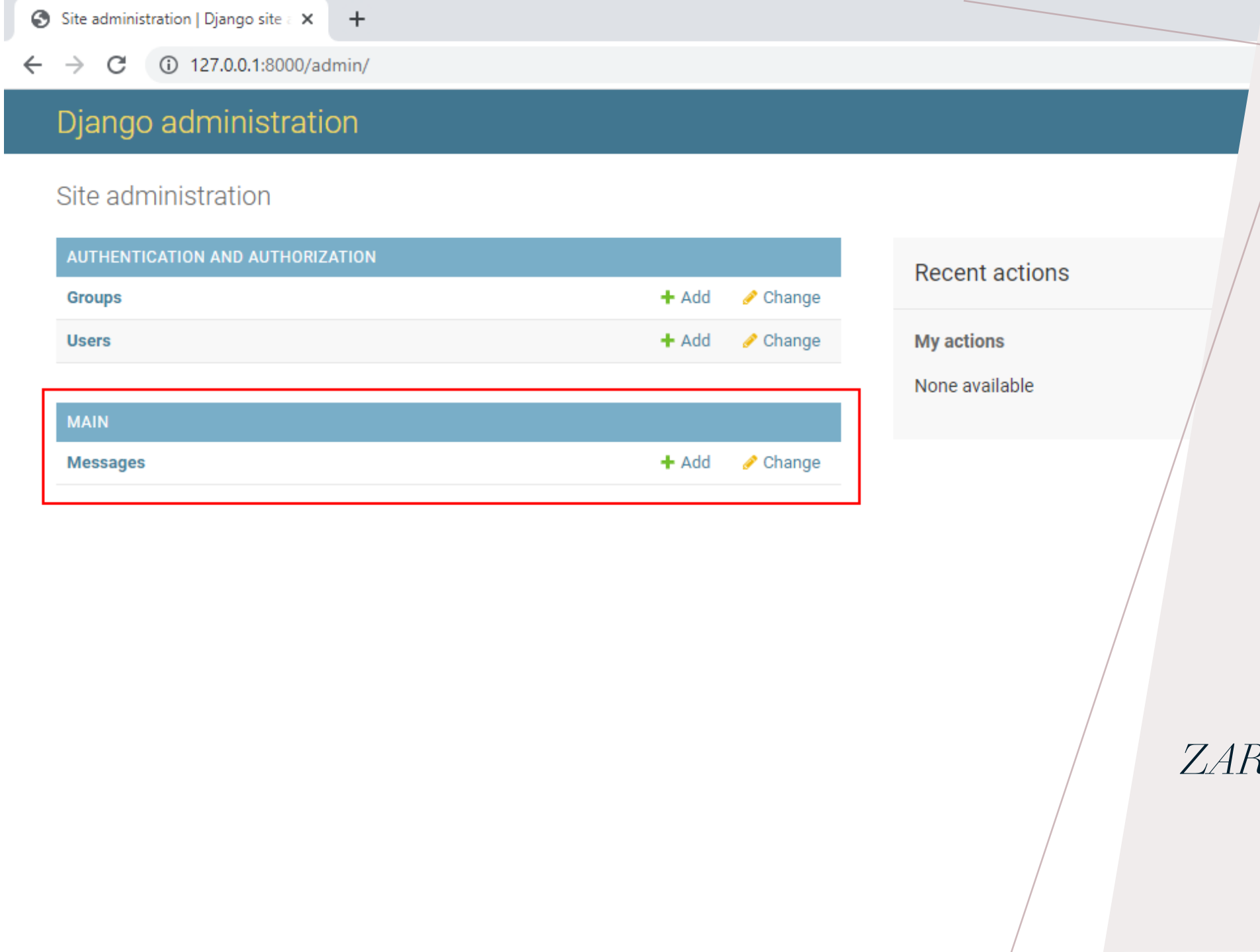
Log in

# LOGOWANIE DO PANELU ADMINISTRATORA

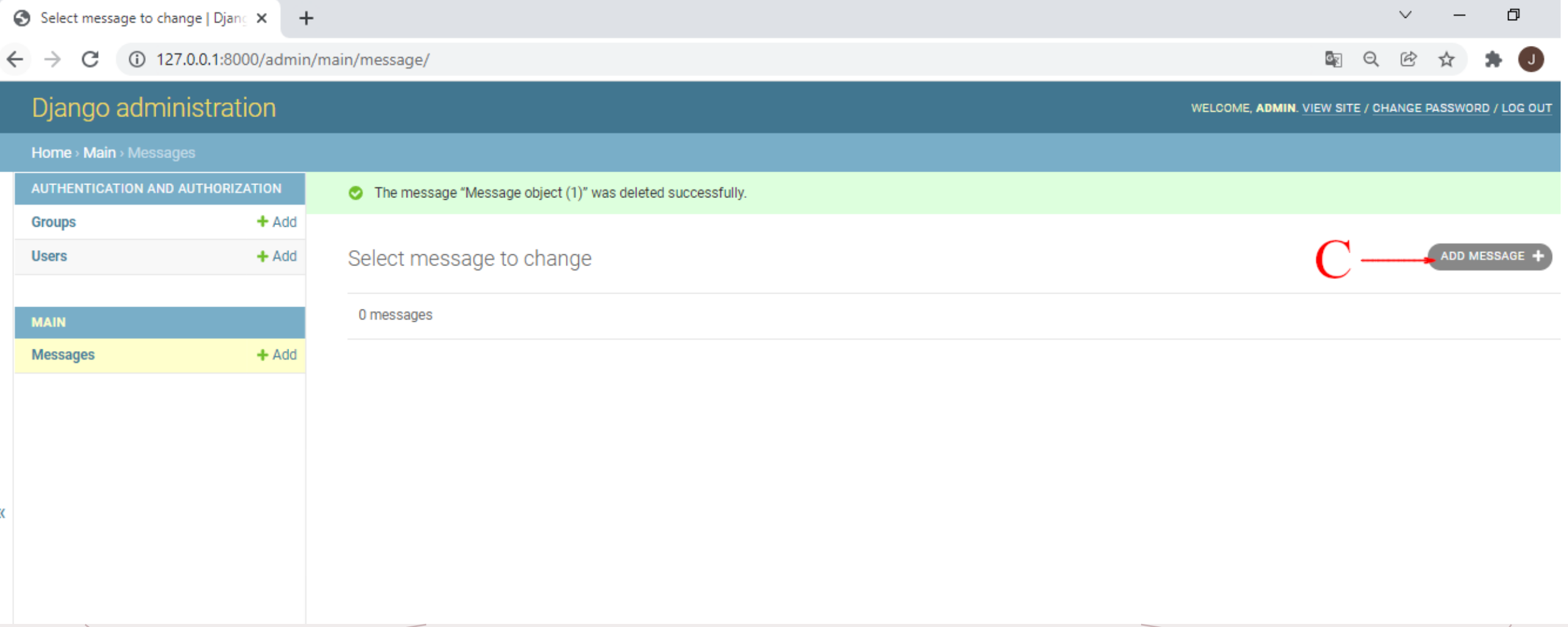



# *PANEL ADMINISTRATORA*

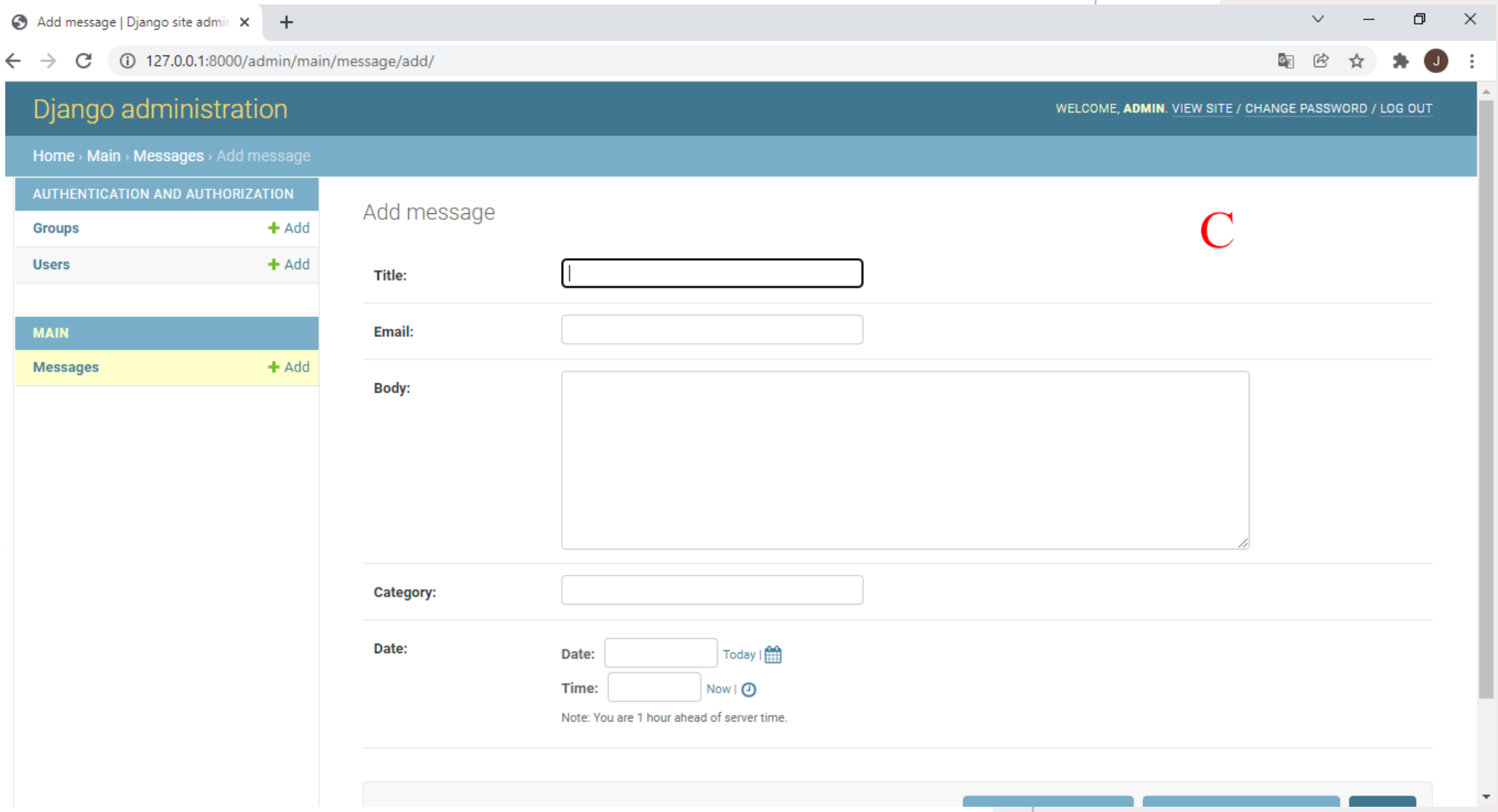

# REJESTROWANIE MODELU W PANELU ADMINISTRATORA



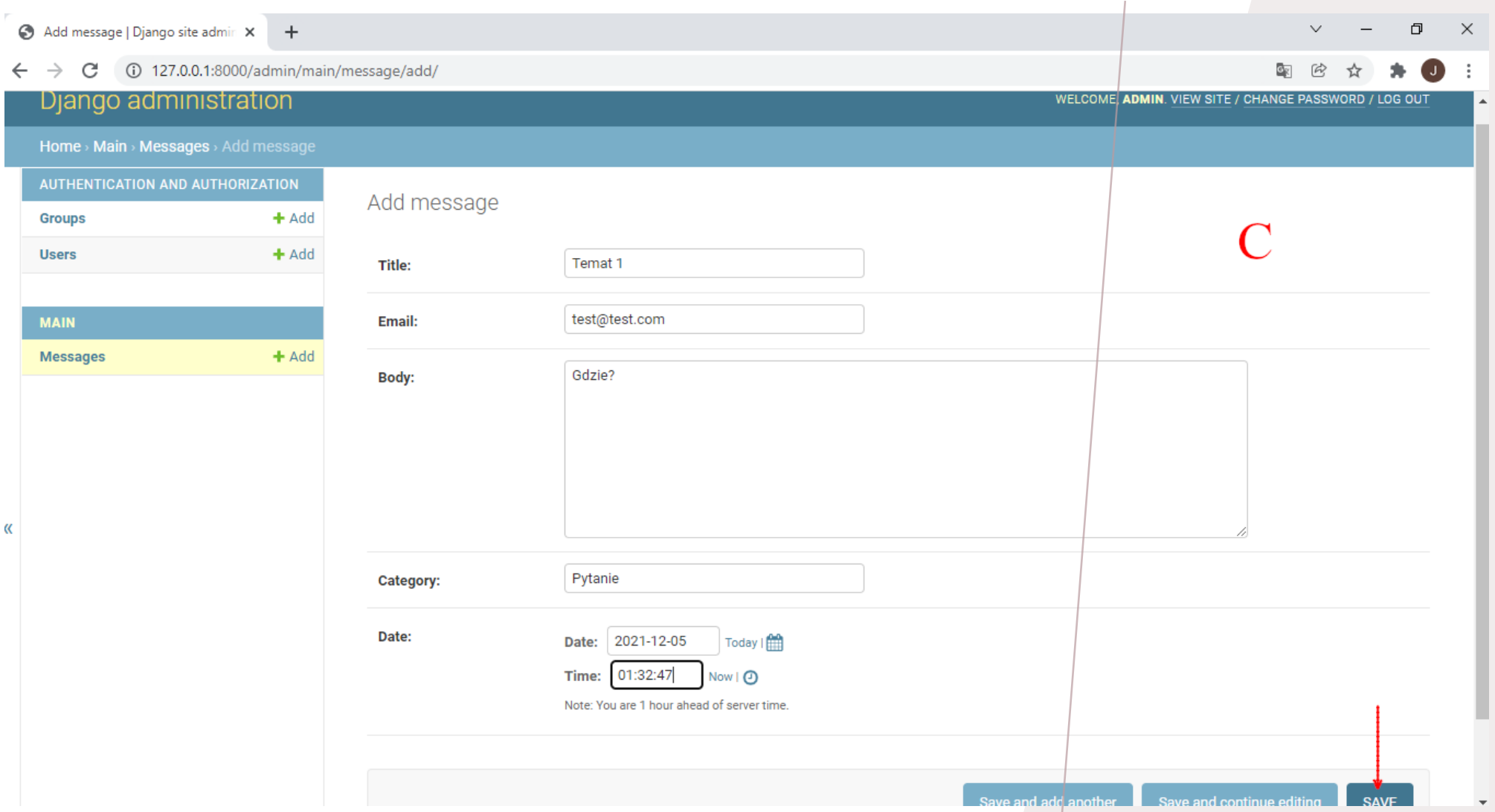

*ZAREJESTROWANY MODEL*



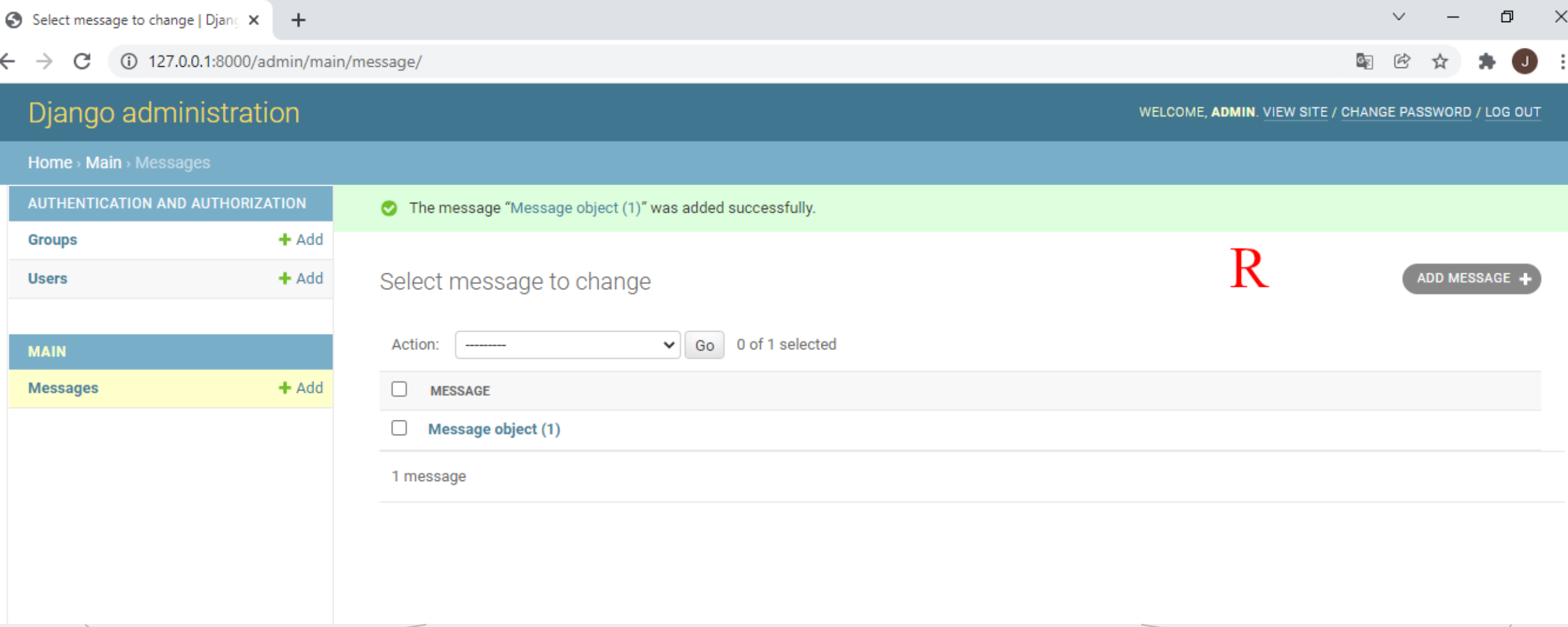

# C (CRUD) W PANELU ADMINISTRATORA



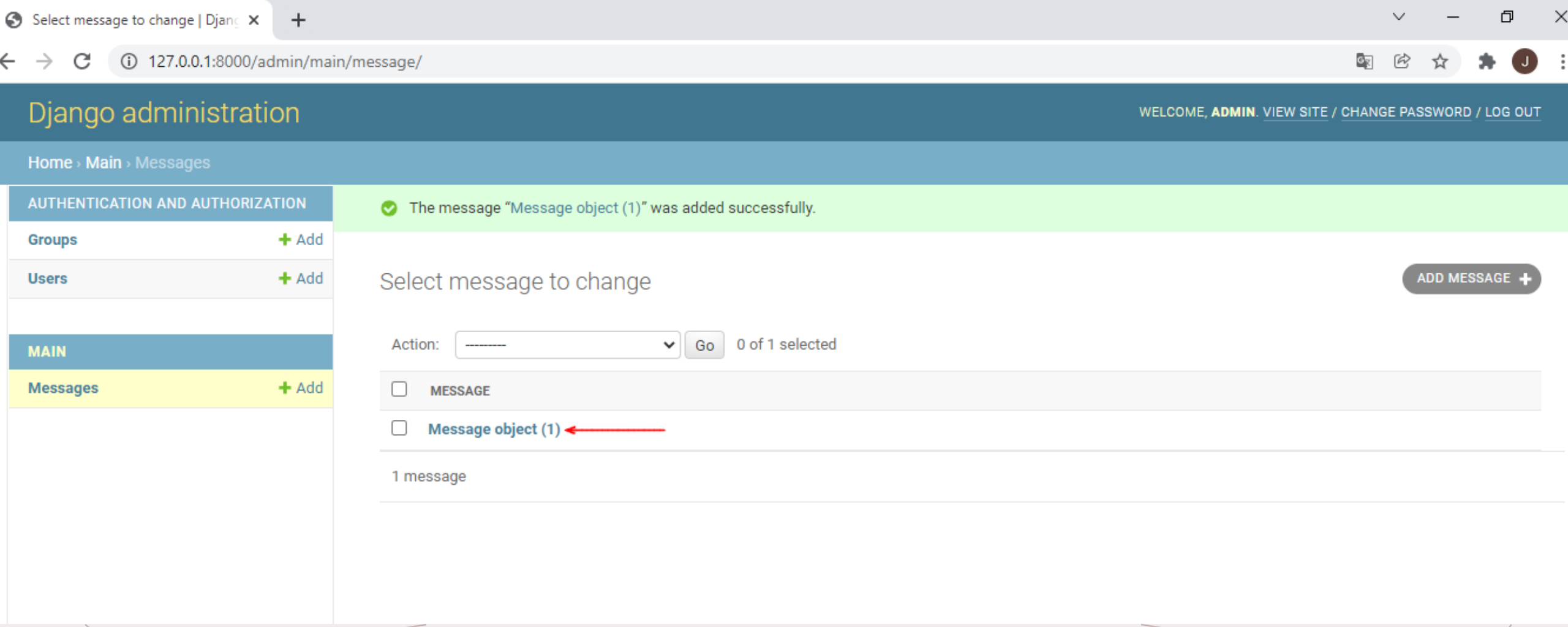

# C (CRUD) W PANELU ADMINISTRATORA



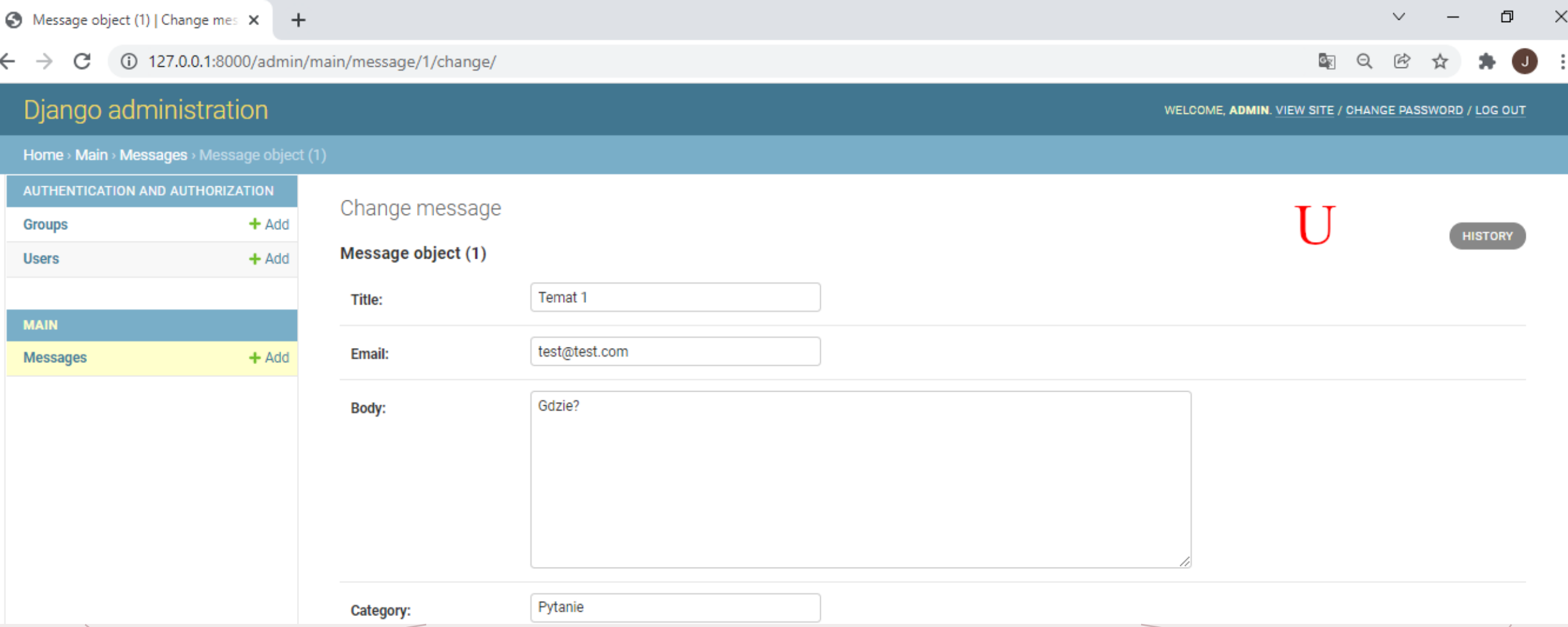

# C (CRUD) W PANELU ADMINISTRATORA



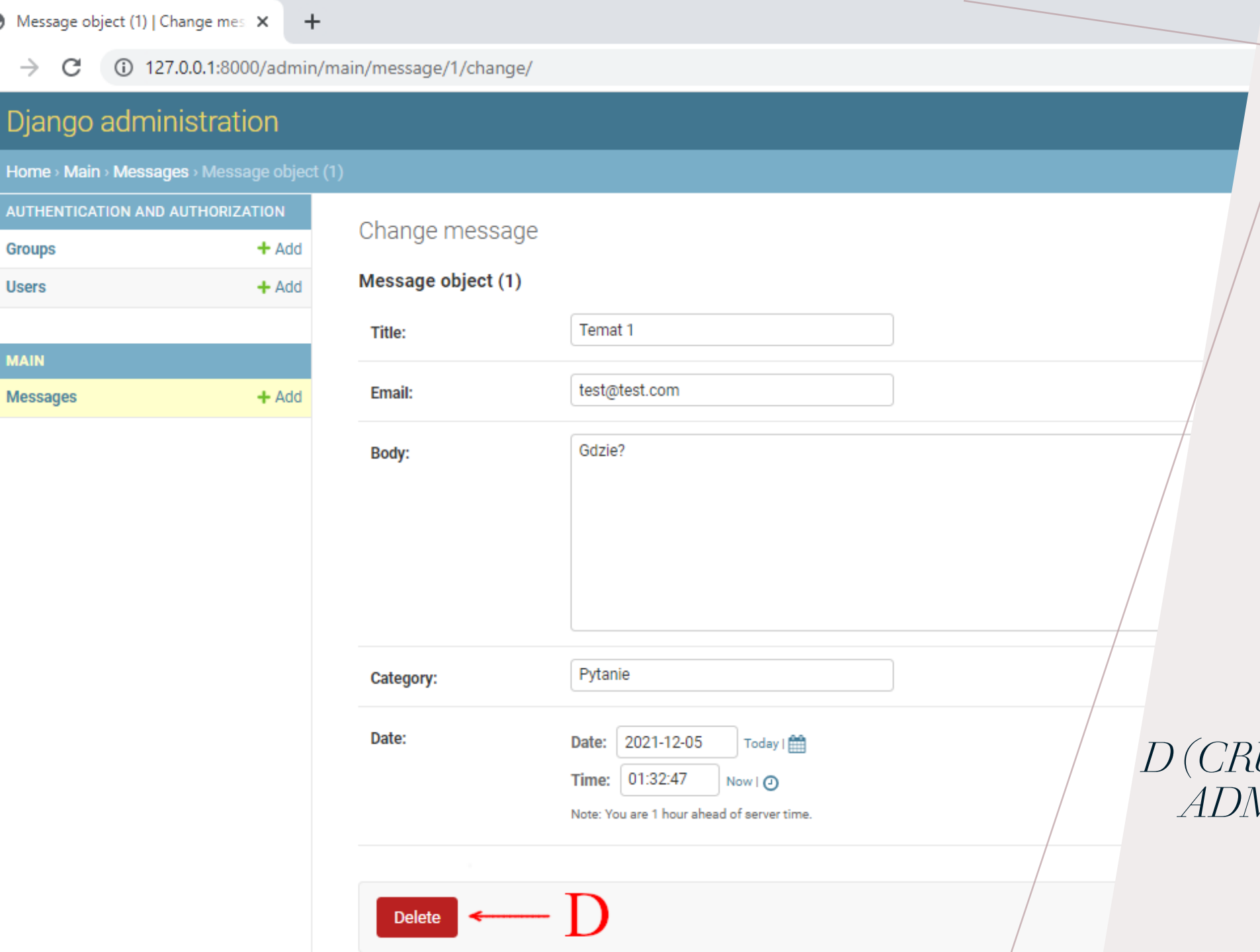

# *R (CRUD) W PANELU ADMINISTRATORA*



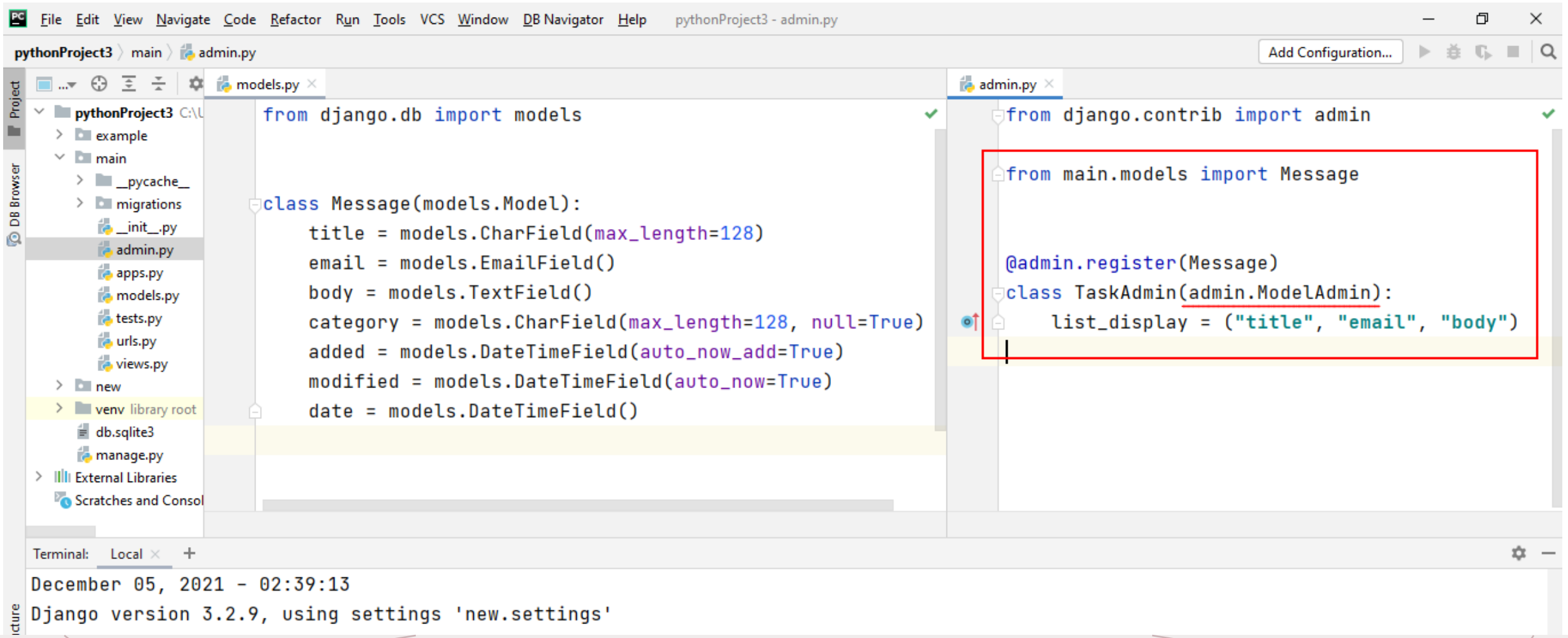

# U (CRUD) W PANELU ADMINISTRATORA



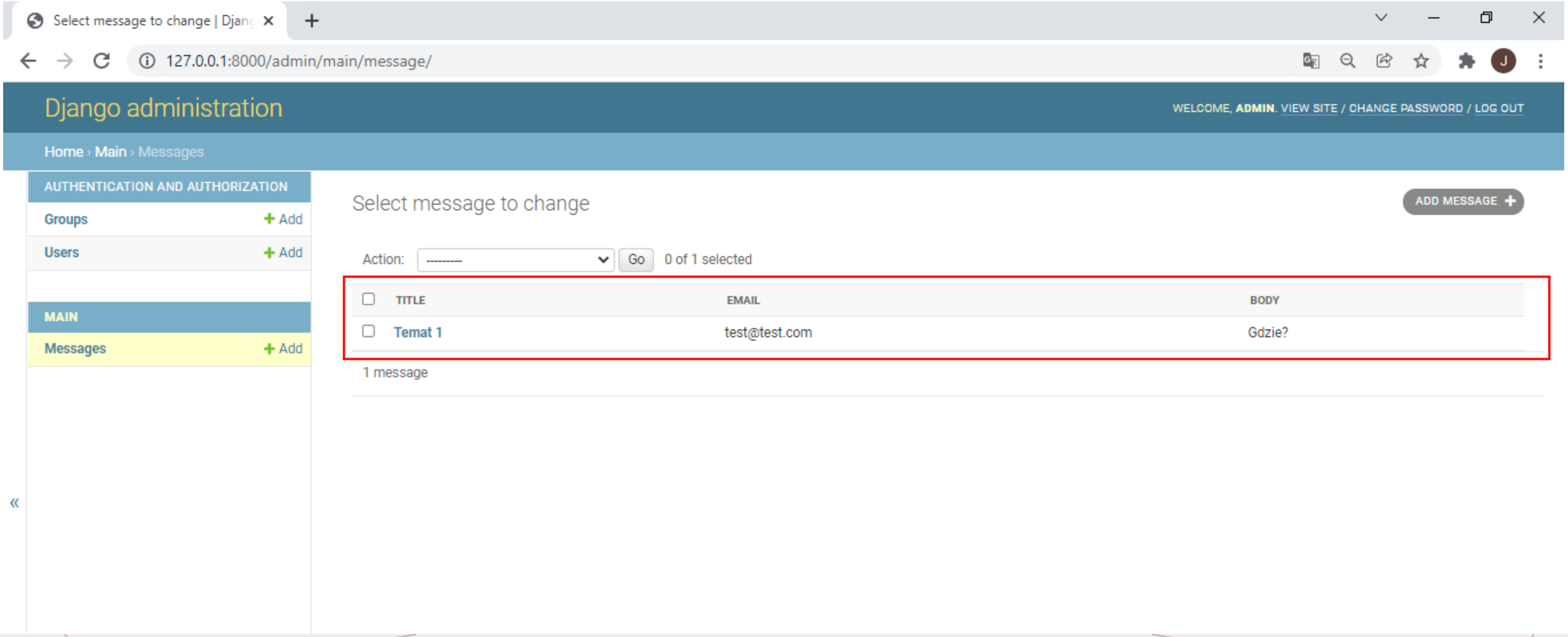

# U (CRUD) W PANELU ADMINISTRATORA




# D (CRUD) W PANELU ADMINISTRATORA




# MODYFIKACJA PANELU ADMINISTRATORA




# *MODYFIKACJA PANELU ADMINISTRATORA*

# DJANGO ORM

## APLIKACJA CRUD

# *APLIKACJA*

# *PYTANIA*

# *PYTANIE I*

Po jakiej klasie musi dziedziczyć klasa, by stała się modelem w Django?

A. DRow
B. Domain
C. models.Model
D. Entity

# *PYTANIE II*

Czym są migracje w Django?

A. Metodyką aktualizacji Django i jego bibliotek do nowszej wersji, niekompatybilnej z poprzednio używaną.

B. Metodą zarządzania danymi użytkowników: prawami, przynależnością, loginem, hasłem, itd.

C. Sposobem na przeniesienie kodu z projektu napisanego we Flasku do Django.

D. Mechanizmem pozwalającym na propagację zmian w modelach na zmiany w bazie danych.

# *PYTANIE III*

W jaki sposób należy dodawać nowe pole do modelu Django ?

A. Zdefiniuj odpowiedni atrybut w modelu, samodzielnie utwórz migracje, a następnie wykonaj komendę 'python manage.py migrate'.

B. Zdefiniuj odpowiedni atrybut w modelu, zapisz modyfikujące zapytanie sql w istniejącej migracji, a następnie wykonaj komendę 'python manage.py migrate'.

C. Zdefiniuj odpowiedni atrybut w modelu i ręcznie zmodyfukuj schamat w tabeli przy użyciu klienta bazodanowego.

D. Zdefiniuj odpowiedni atrybut w modelu, a następnie wykonaj komendy: 'python manage.py makemigrations', 'python manage.py migrate'

# *PYTANIE IV*

Czym są migracje Django?

A. Migracje to pliki, w których Django przechowuje zmiany modelu.
B. Migracje to pliki przechowywane w folderze migrations.
C. Migracje to pliki tworzone komendą python manage.py makemigrations.
D. Wsystkie powyższe odpowiedzi są prawidłowe.

# *PYTANIE V*

Pod jakim adresem znajduje się panel admina Django?

A. /login
B. /admin
C. /administrator
D. /django-admin

# *PYTANIE VI*

Kiedy używamy funkcji HttpResponseRedirect ?

A. Nie istnieje coś takiego.
B. Kiedy dołączamy link do elementu html typu button.
C. W momencie, kiedy pracujemy z danymi wysyłanymi metodą GET.
D. W momencie, kiedy pracujemy z danymi wysyłanymi metodą POST.

# *PYTANIE VII*

W którym pliku zawarte są informacje potrzebne do połączenia się z bazą danych?

A. database.py
B. settings.py
C. set.py
D. connector.py

# *PYTANIE VIII*

Komenda migrate musi zostać wykonana zawsze kiedy tworzymy nową aplikację Django.

A. Prawda.

B. Fałsz.

# *PYTANIE IX*

Które z poniższych stwierdzeń jest prawdziwe dla modelu?

A. Musimy samodzielnie wskazać miejsce przechowywania modeli, ponieważ Django nie zapewnia wsparcia dla bazy danych.

B. Modele są źródłem informacji o danych aplikacji.

C. We wzorcu MTV, widok decyduje jakie dane są prezentowane, a model decyduje w jaki sposób będą się one wyświetlały na stronie.

D. Deweloperzy Django mogą komunikować się z bazą danych bezpośrednio bez jakiegokolwiek interfejsu czy modelu.

# *PYTANIE X*

Po jakiej klasie musimy dziedziczyć, żeby zmienić widok formularzy w panelu admina Django?

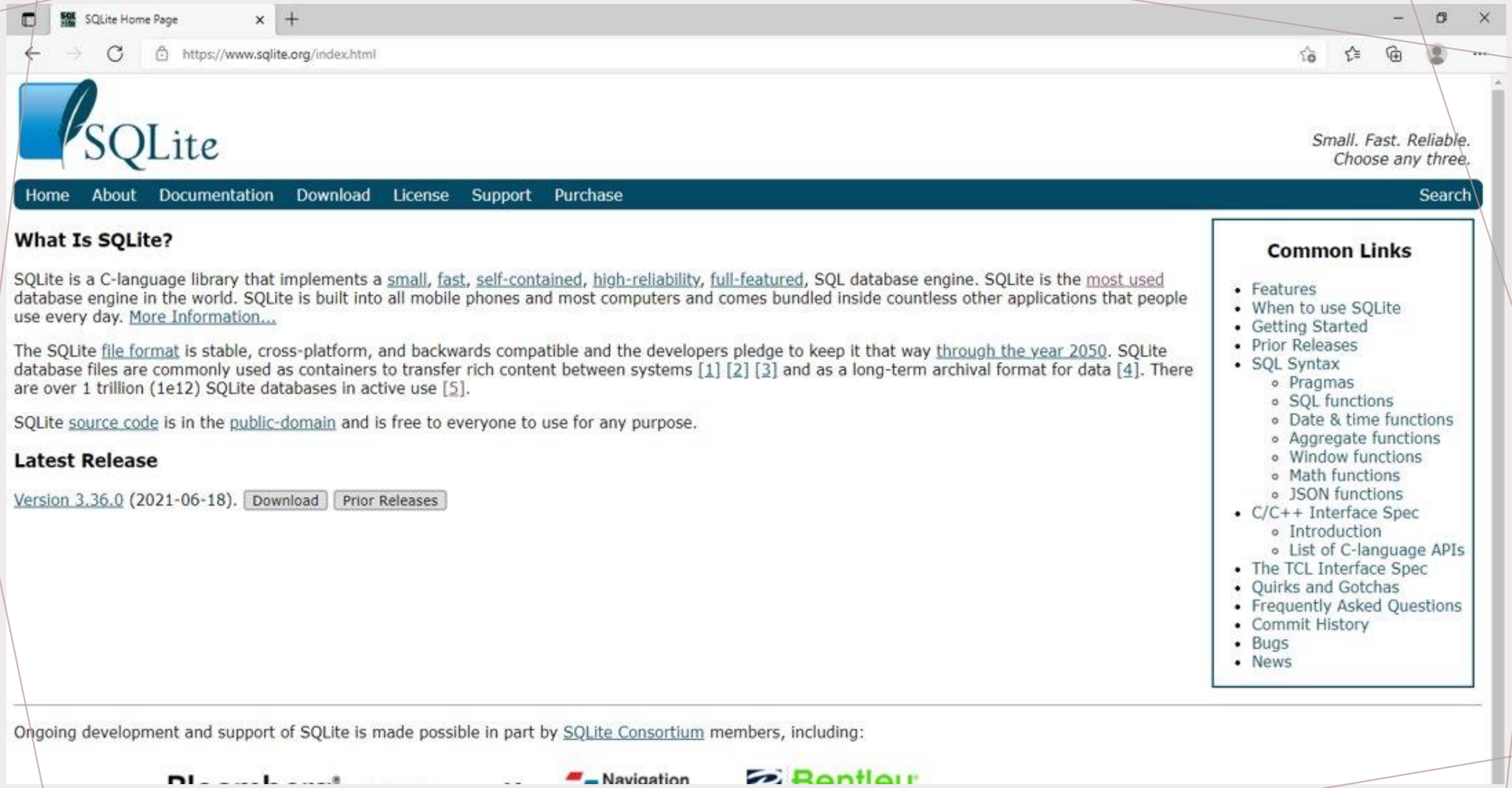
A. AdminModel
B. admin.ModelAdmin
C. AdministrationModel
D. admin.Model

# *DODATKI*

# *DBMS*

*DataBase Management System*

SQLite


SQLite Home Page
https://www.sqlite.org/index.html
SQLite
Small. Fast. Reliable.
Choose any three.
Home About Documentation Download License Support Purchase Search
What Is SQLite?
SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day. More Information...
The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems [1] [2] [3] and as a long-term archival format for data [4]. There are over 1 trillion (1e12) SQLite databases in active use [5].
SQLite source code is in the public-domain and is free to everyone to use for any purpose.
Latest Release
Version 3.36.0 (2021-06-18). Download Prior Releases
Common Links
Features
When to use SQLite
Getting Started
Prior Releases
SQL Syntax
Pragmas
SQL functions
Date & time functions
Aggregate functions
Window functions
Math functions
JSON functions
C/C++ Interface Spec
Introduction
List of C-language APIs
The TCL Interface Spec
Quirks and Gotchas
Frequently Asked Questions
Commit History
Bugs
News
Ongoing development and support of SQLite is made possible in part by SQLite Consortium members, including:

# SQLite Tutorial

This **SQLite tutorial** teaches you everything you need to know to start using SQLite effectively. In this tutorial, you will learn SQLite step by step through extensive hands-on practices.

This SQLite tutorial is designed for developers who want to use SQLite as the back-end database or to use SQLite to manage structured data in applications including desktop, web, and mobile apps.

SQLite is an open-source, zero-configuration, self-contained, stand-alone, transaction relational database engine designed to be embedded into an application.

## Getting started with SQLite

You should go through this section if this is the first time you have worked with SQLite. Follow these 4-easy steps to get started with SQLite fast.

- First, help you answer the first and important question: what is SQLite? You will have a brief overview of SQLite.
- Second, show you step by step how to download and install the SQLite tools on your computer.
- Third, introduce you to an SQLite sample database and walk you through the steps of using the sample database for practicing.
- Finally, guide you on how to use the sqlite3 commands.

## Basic SQLite tutorial

This section presents basic SQL statements that you can use with SQLite. You will first start querying data from the sample database. If you are already familiar with SQL, you will notice the differences between SQL standard and SQL dialect used in SQLite.

### Section 1. Simple query

Search ...

GETTING STARTED

What Is SQLite

Download & Install SQLite

SQLite Sample Database

SQLite Commands



SQLITE TUTORIAL

SQLite Select

SQLite Order By

SQLite Select Distinct

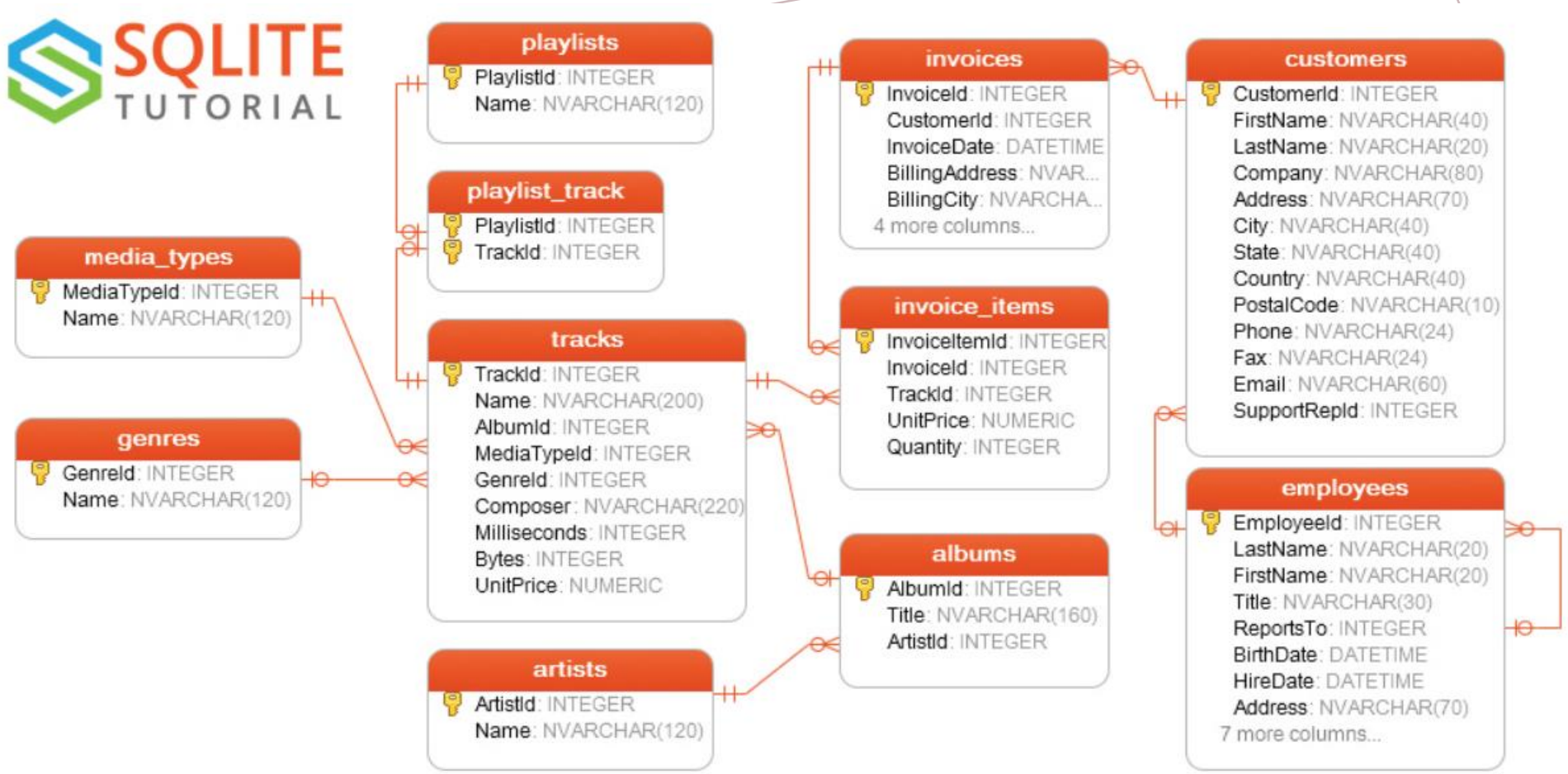
SQLite Where

SQLite Limit

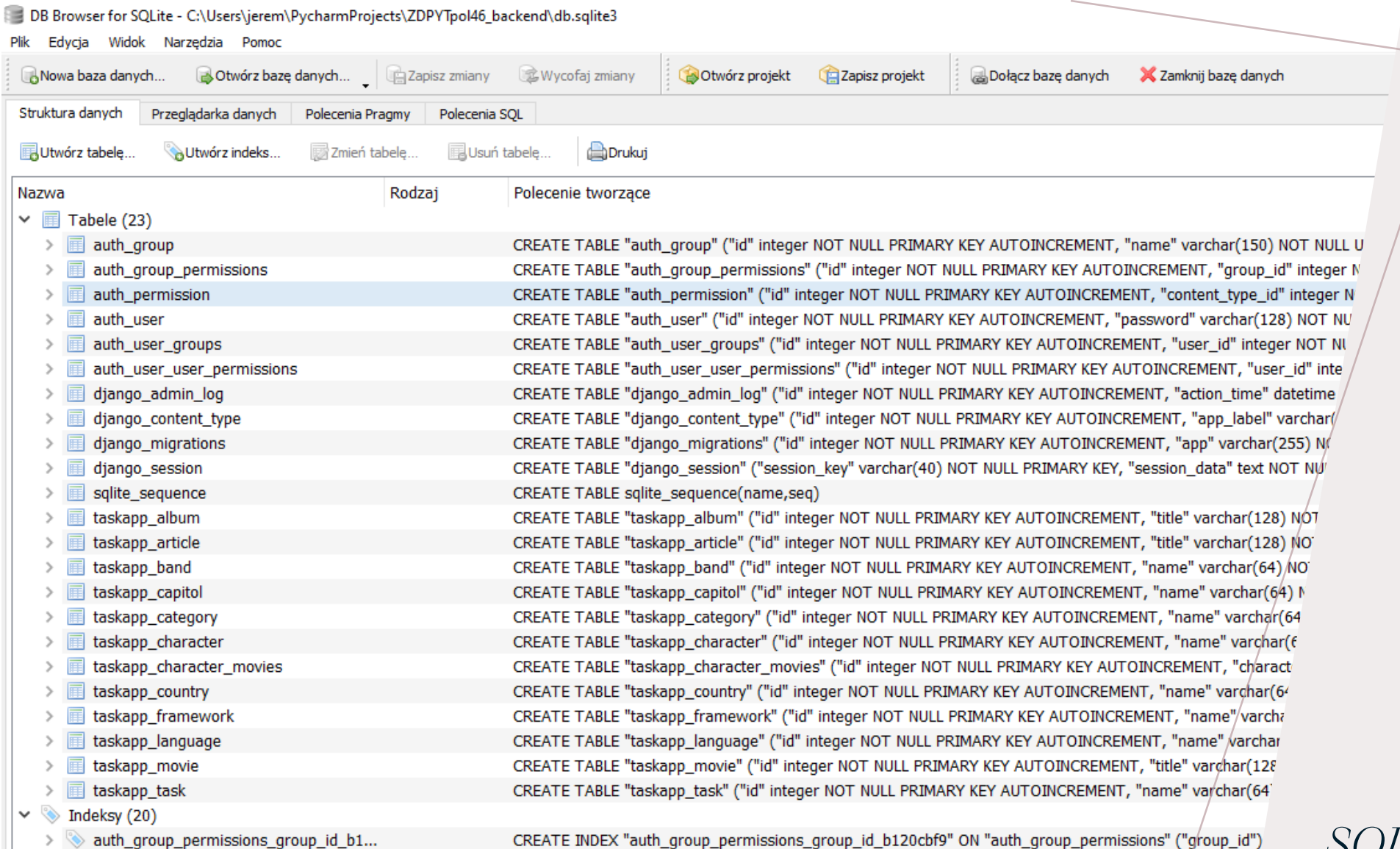
SQLite BETWEEN

SQLite IN

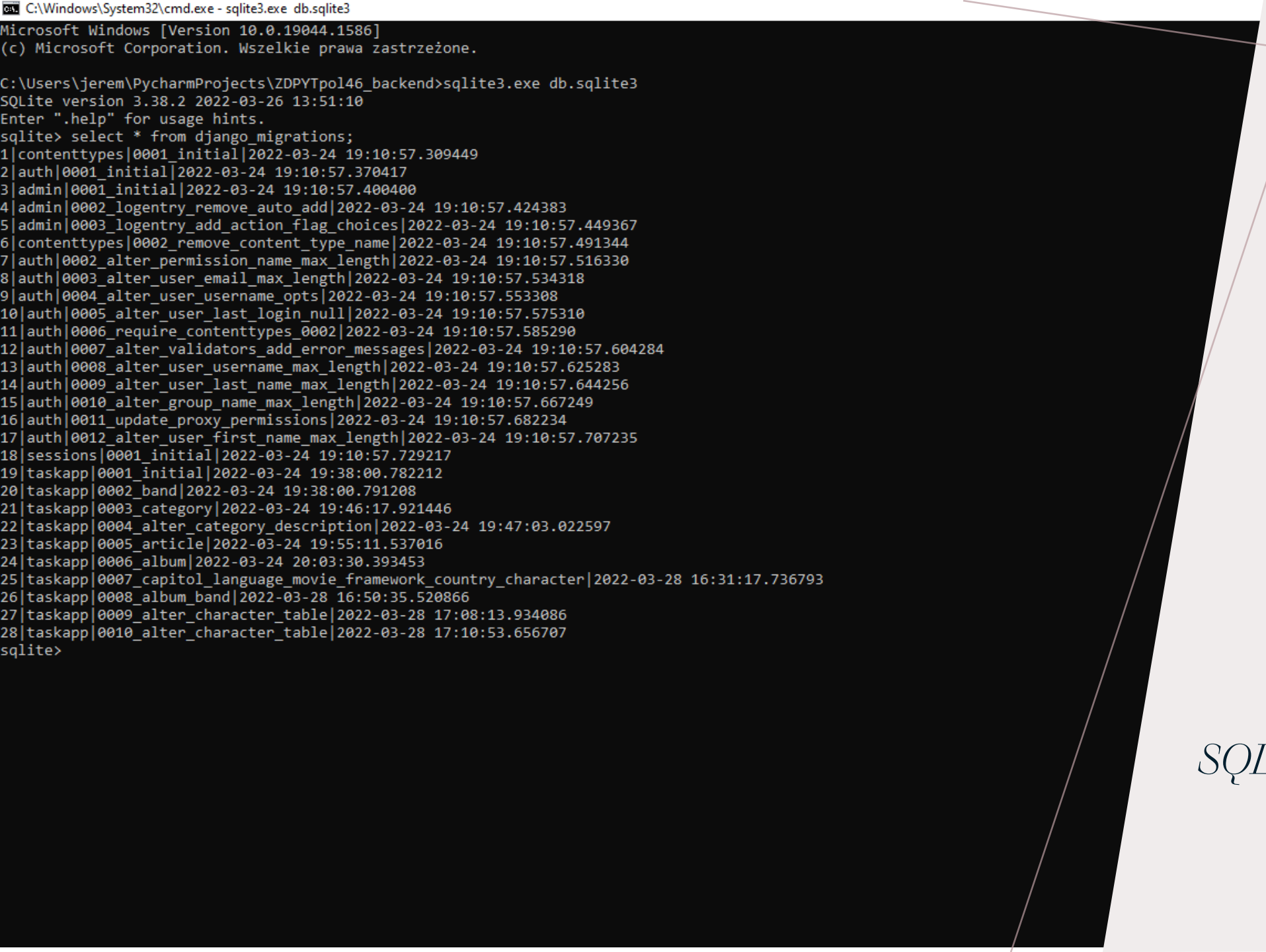
SQLite Like


SQLITE TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

# CHINOOK




# *SQLITE GUI CLIENT – DB BROWSER*




# *SQLITE CLI CLIENT – SQLITE3*

MySQL

MySQL

mysql.com

The world's most popular open source database

Contact MySQL | Login | Register

MYSQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE

Products Cloud Services Partners Customers Why MySQL? News & Events How to Buy


## Try MySQL HeatWave for Free

Get US$300 in credit for 30 days

TRY NOW

### MySQL HeatWave

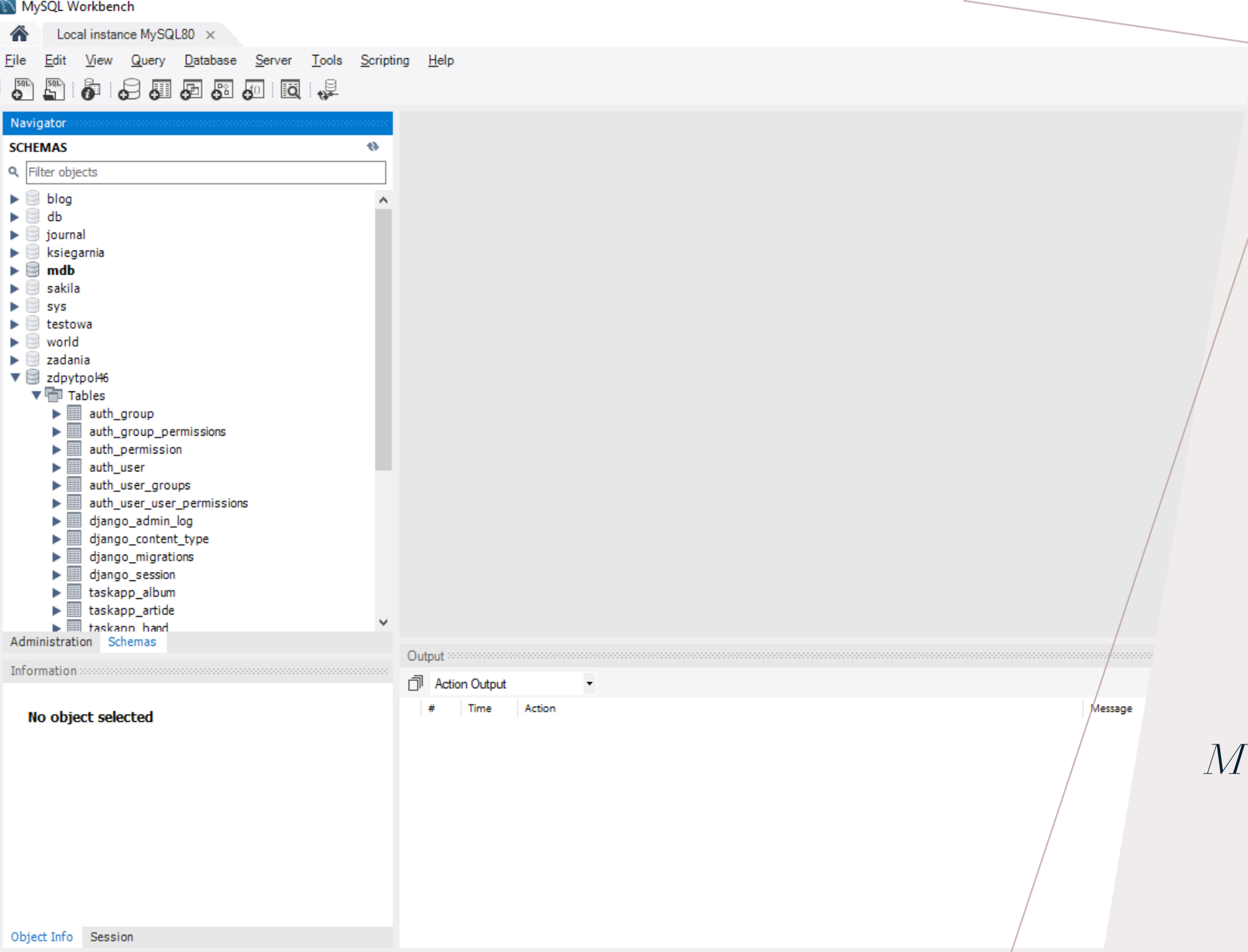
MySQL HeatWave is a fully managed service that enables customers to run OLTP and OLAP workloads directly from their MySQL Database. HeatWave, an integrated, high-performance query accelerator, boosts MySQL performance by 5400x.

Learn More »

### MySQL Enterprise Edition

The most comprehensive set of advanced features, management tools and technical support to achieve the highest levels of MySQL scalability, security, reliability, and uptime.

Learn More »



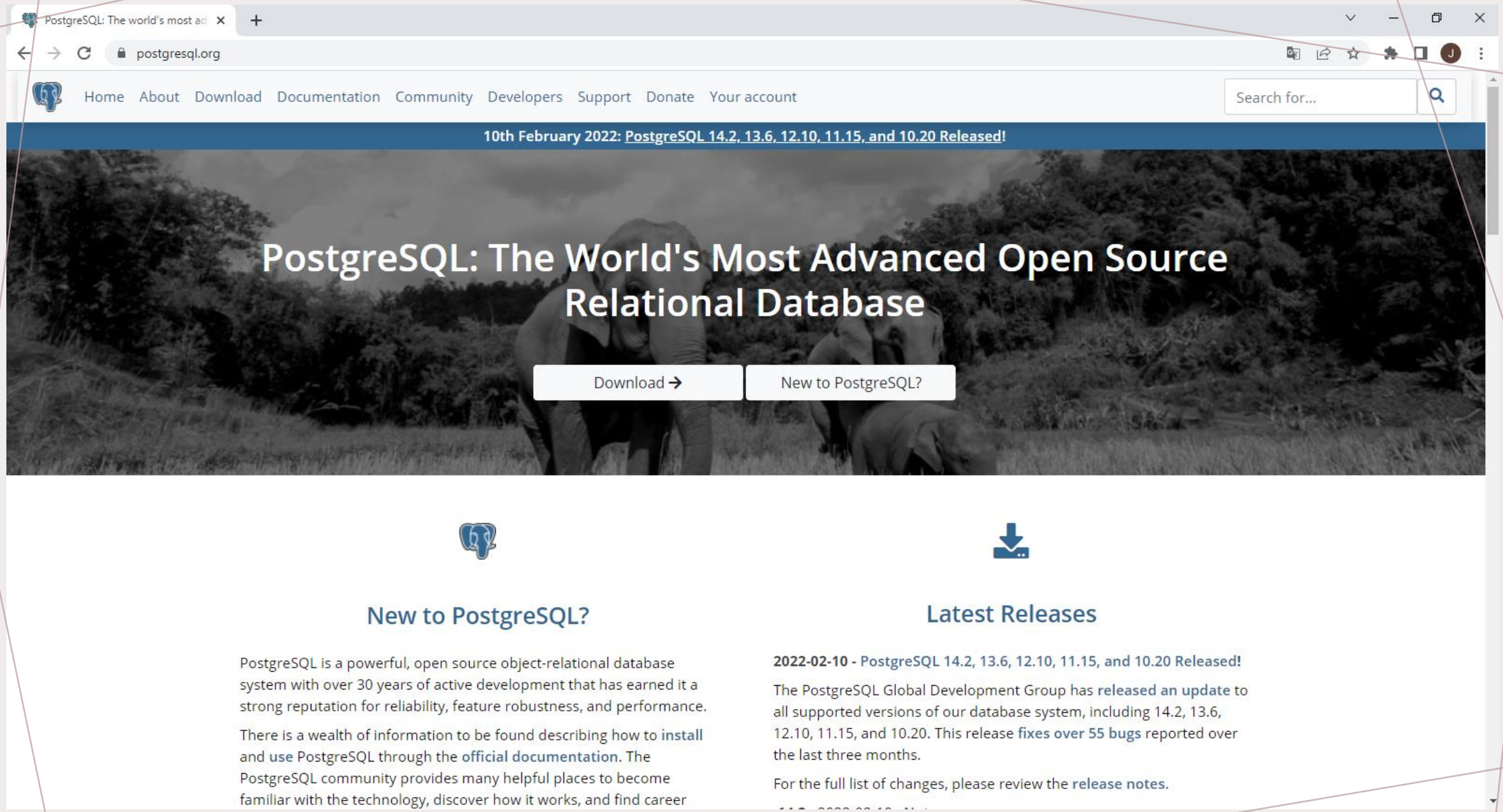

# MYSQL GUI CLIENT - WORKBENCH

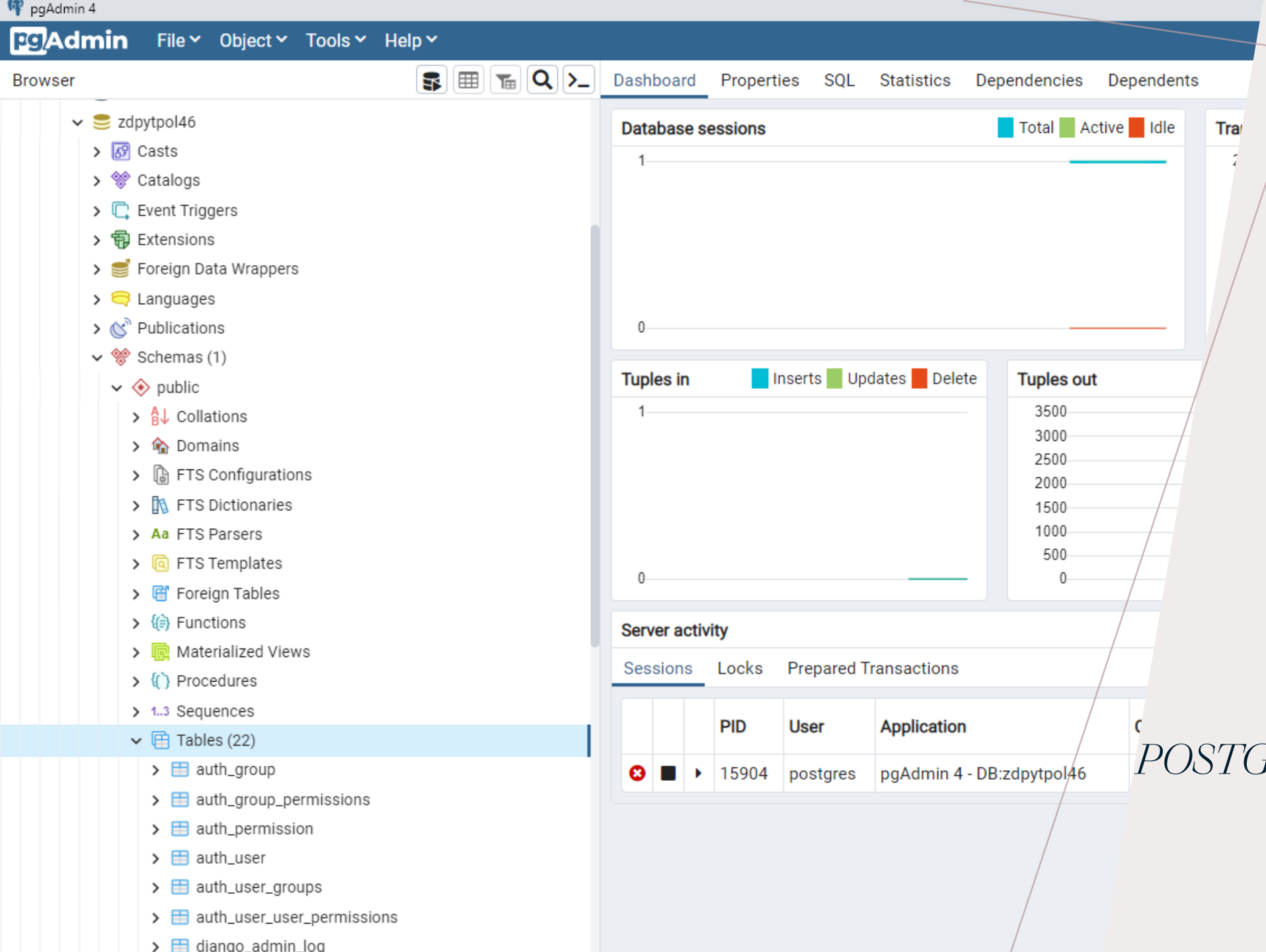
Wiersz polecenia - mysql -u root -p

```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\jerem>mysql -u root -p
Enter password: ************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from zdpytpol46.django_migrations;
+----+--------------+----------------------------------------------------------+----------------------------+
| id | app          | name                                                     | applied                    |
+----+--------------+----------------------------------------------------------+----------------------------+
|  1 | contenttypes | 0001_initial                                             | 2022-03-28 17:45:13.807284 |
|  2 | auth         | 0001_initial                                             | 2022-03-28 17:45:14.926662 |
|  3 | admin        | 0001_initial                                             | 2022-03-28 17:45:15.208500 |
|  4 | admin        | 0002_logentry_remove_auto_add                            | 2022-03-28 17:45:15.227488 |
|  5 | admin        | 0003_logentry_add_action_flag_choices                    | 2022-03-28 17:45:15.244480 |
|  6 | contenttypes | 0002_remove_content_type_name                            | 2022-03-28 17:45:15.560297 |
|  7 | auth         | 0002_alter_permission_name_max_length                    | 2022-03-28 17:45:15.674246 |
|  8 | auth         | 0003_alter_user_email_max_length                         | 2022-03-28 17:45:15.710210 |
|  9 | auth         | 0004_alter_user_username_opts                            | 2022-03-28 17:45:15.726201 |
| 10 | auth         | 0005_alter_user_last_login_null                          | 2022-03-28 17:45:15.839137 |
| 11 | auth         | 0006_require_contenttypes_0002                           | 2022-03-28 17:45:15.844132 |
| 12 | auth         | 0007_alter_validators_add_error_messages                 | 2022-03-28 17:45:15.858125 |
| 13 | auth         | 0008_alter_user_username_max_length                      | 2022-03-28 17:45:15.989051 |
| 14 | auth         | 0009_alter_user_last_name_max_length                     | 2022-03-28 17:45:16.086009 |
| 15 | auth         | 0010_alter_group_name_max_length                         | 2022-03-28 17:45:16.125987 |
| 16 | auth         | 0011_update_proxy_permissions                            | 2022-03-28 17:45:16.140977 |
| 17 | auth         | 0012_alter_user_first_name_max_length                    | 2022-03-28 17:45:16.255906 |
| 18 | sessions     | 0001_initial                                             | 2022-03-28 17:45:16.308865 |
| 19 | taskapp      | 0001_initial                                             | 2022-03-28 17:45:16.354856 |
| 20 | taskapp      | 0002_band                                                | 2022-03-28 17:45:16.393835 |
| 21 | taskapp      | 0003_category                                            | 2022-03-28 17:45:16.438843 |
| 22 | taskapp      | 0004_alter_category_description                          | 2022-03-28 17:45:16.500810 |
| 23 | taskapp      | 0005_article                                             | 2022-03-28 17:45:16.542807 |
| 24 | taskapp      | 0006_album                                               | 2022-03-28 17:45:16.596777 |
| 25 | taskapp      | 0007_capitol_language_movie_framework_country_character  | 2022-03-28 17:45:17.217439 |
| 26 | taskapp      | 0008_album_band                                          | 2022-03-28 17:45:17.330310 |
+----+--------------+----------------------------------------------------------+----------------------------+
26 rows in set (0.00 sec)

mysql>
```

# *MYSQL CLI CLIENT - MYSQL*

PostgreSQL


PostgreSQL: The world's most ad
postgresql.org
Home About Download Documentation Community Developers Support Donate Your account
Search for...
10th February 2022: PostgreSQL 14.2, 13.6, 12.10, 11.15, and 10.20 Released!
PostgreSQL: The World's Most Advanced Open Source Relational Database
Download →
New to PostgreSQL?
New to PostgreSQL?
PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.
There is a wealth of information to be found describing how to install and use PostgreSQL through the official documentation. The PostgreSQL community provides many helpful places to become familiar with the technology, discover how it works, and find career
Latest Releases
2022-02-10 - PostgreSQL 14.2, 13.6, 12.10, 11.15, and 10.20 Released!
The PostgreSQL Global Development Group has released an update to all supported versions of our database system, including 14.2, 13.6, 12.10, 11.15, and 10.20. This release fixes over 55 bugs reported over the last three months.
For the full list of changes, please review the release notes.




# POSTGRESQL GUI CLIENT - PGADMIN

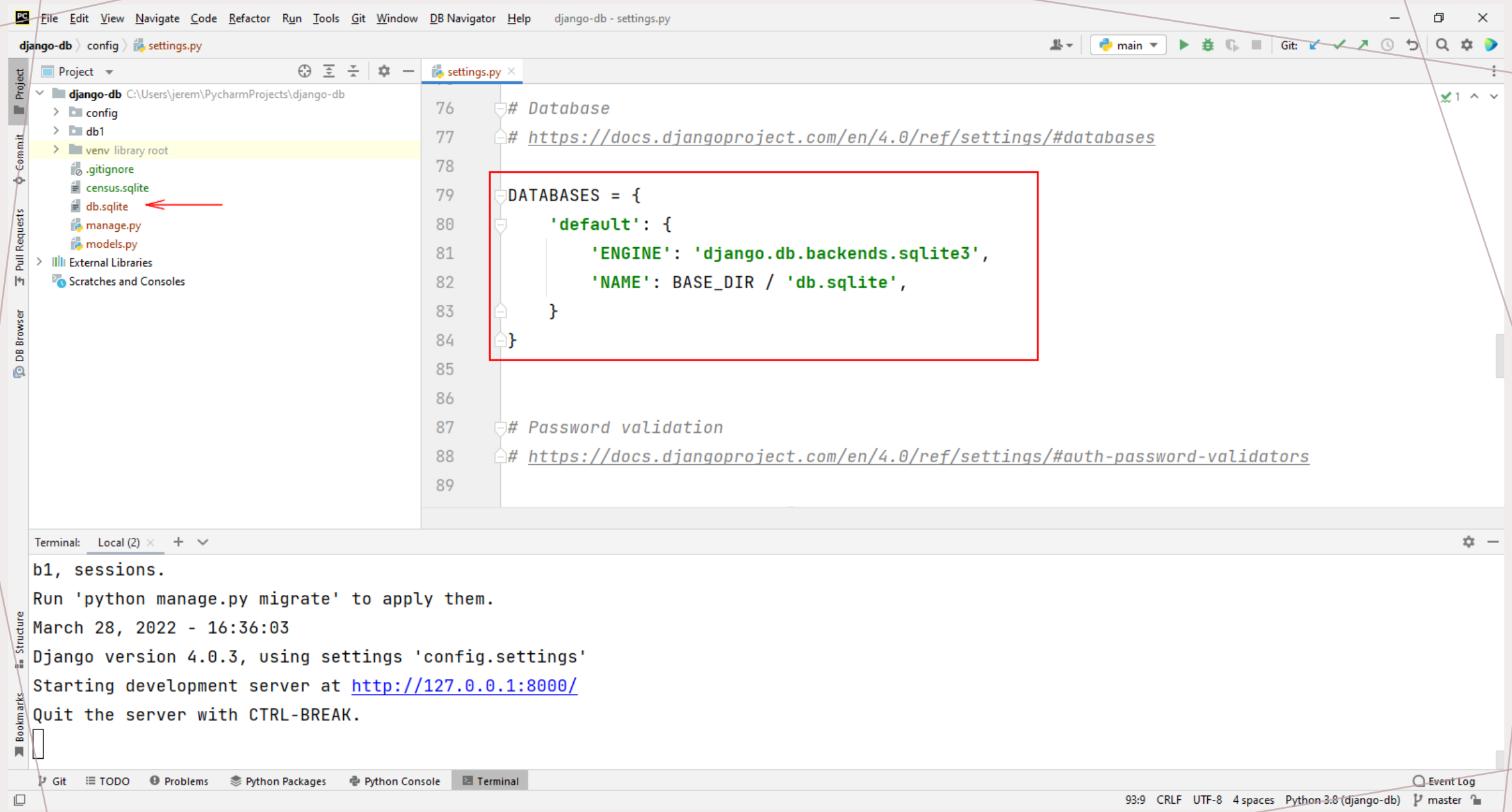
SQL Shell (psql)

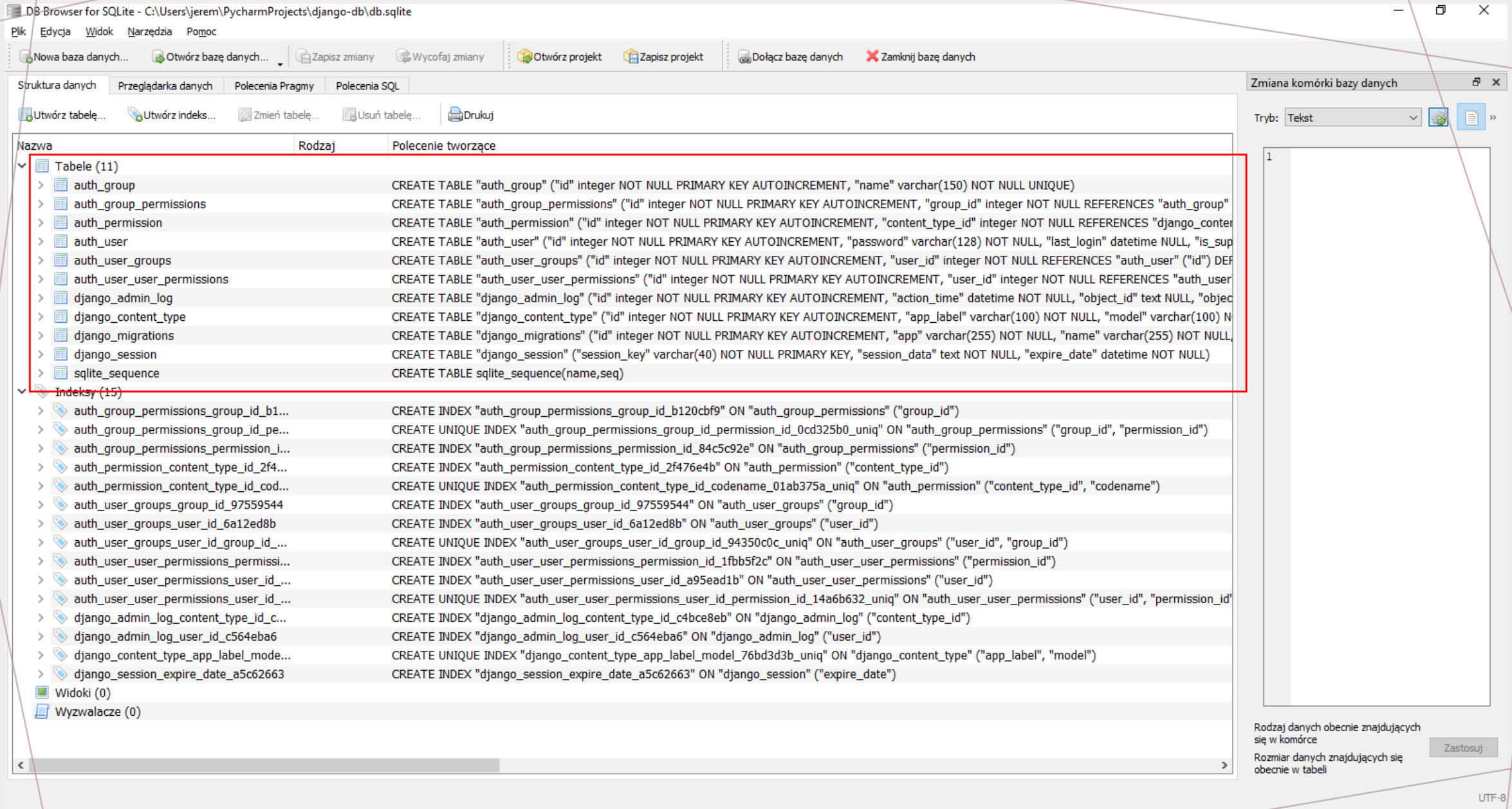
```
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.1)
WARNING: Console code page (852) differs from Windows code page (1250)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \c zdpytpol46
You are now connected to database "zdpytpol46" as user "postgres".
zdpytpol46=# select * from django_migrations;
 id |     app      |                              name                              |              applied
----+--------------+----------------------------------------------------------------+-------------------------------
  1 | contenttypes | 0001_initial                                                   | 2022-03-28 19:28:11.83978+02
  2 | auth         | 0001_initial                                                   | 2022-03-28 19:28:12.010683+02
  3 | admin        | 0001_initial                                                   | 2022-03-28 19:28:12.050658+02
  4 | admin        | 0002_logentry_remove_auto_add                                  | 2022-03-28 19:28:12.068648+02
  5 | admin        | 0003_logentry_add_action_flag_choices                          | 2022-03-28 19:28:12.07964+02
  6 | contenttypes | 0002_remove_content_type_name                                  | 2022-03-28 19:28:12.116621+02
  7 | auth         | 0002_alter_permission_name_max_length                          | 2022-03-28 19:28:12.128614+02
  8 | auth         | 0003_alter_user_email_max_length                               | 2022-03-28 19:28:12.142609+02
  9 | auth         | 0004_alter_user_username_opts                                  | 2022-03-28 19:28:12.153598+02
 10 | auth         | 0005_alter_user_last_login_null                                | 2022-03-28 19:28:12.164591+02
 11 | auth         | 0006_require_contenttypes_0002                                 | 2022-03-28 19:28:12.166591+02
 12 | auth         | 0007_alter_validators_add_error_messages                       | 2022-03-28 19:28:12.176586+02
 13 | auth         | 0008_alter_user_username_max_length                            | 2022-03-28 19:28:12.201572+02
 14 | auth         | 0009_alter_user_last_name_max_length                           | 2022-03-28 19:28:12.212564+02
 15 | auth         | 0010_alter_group_name_max_length                               | 2022-03-28 19:28:12.234551+02
 16 | auth         | 0011_update_proxy_permissions                                  | 2022-03-28 19:28:12.245545+02
 17 | auth         | 0012_alter_user_first_name_max_length                          | 2022-03-28 19:28:12.25454+02
 18 | sessions     | 0001_initial                                                   | 2022-03-28 19:28:12.279527+02
 19 | taskapp      | 0001_initial                                                   | 2022-03-28 19:28:12.287522+02
 20 | taskapp      | 0002_band                                                      | 2022-03-28 19:28:12.303513+02
 21 | taskapp      | 0003_category                                                  | 2022-03-28 19:28:12.322503+02
 22 | taskapp      | 0004_alter_category_description                                | 2022-03-28 19:28:12.3295+02
 23 | taskapp      | 0005_article                                                   | 2022-03-28 19:28:12.351487+02
 24 | taskapp      | 0006_album                                                     | 2022-03-28 19:28:12.362479+02
 25 | taskapp      | 0007_capitol_language_movie_framework_country_character        | 2022-03-28 19:28:12.453437+02
 26 | taskapp      | 0008_album_band                                                | 2022-03-28 19:28:12.467426+02
(26 rows)


zdpytpol46=#
```

# POSTGRESQL CLI CLIENT - PSQL

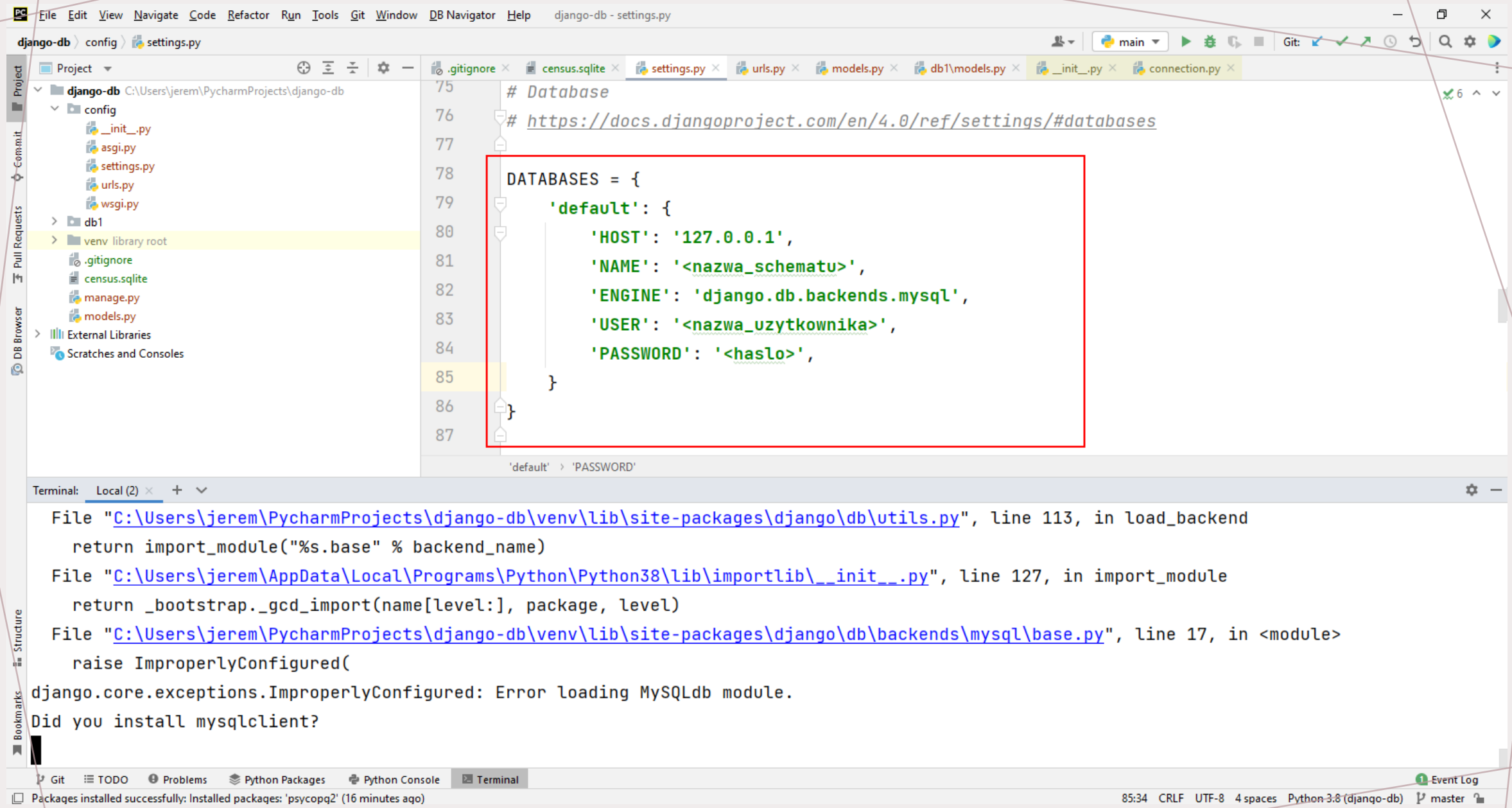
# *KONFIGURACJA POLACZENIA Z BAZA W DJANGO*

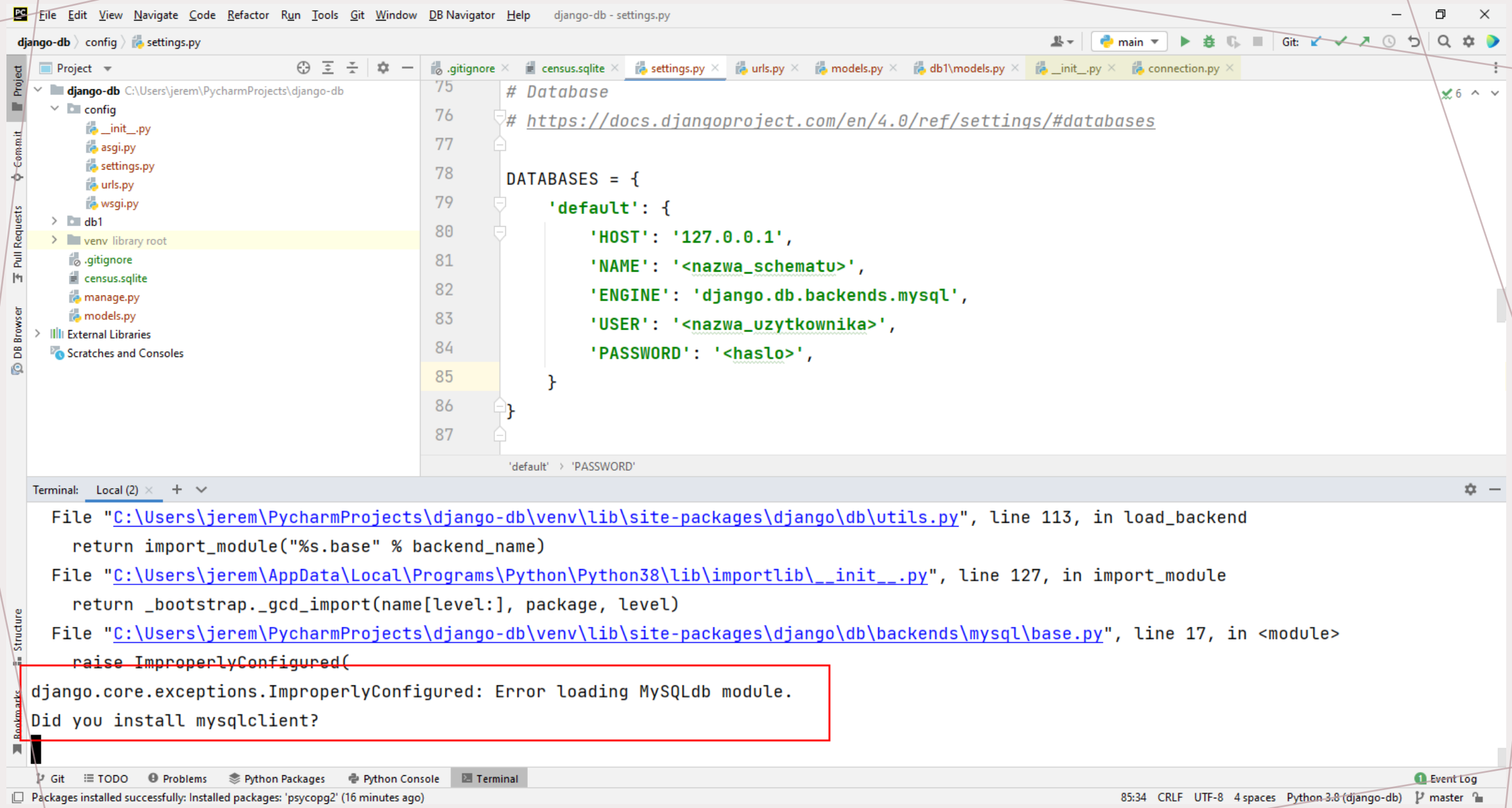
SQLite

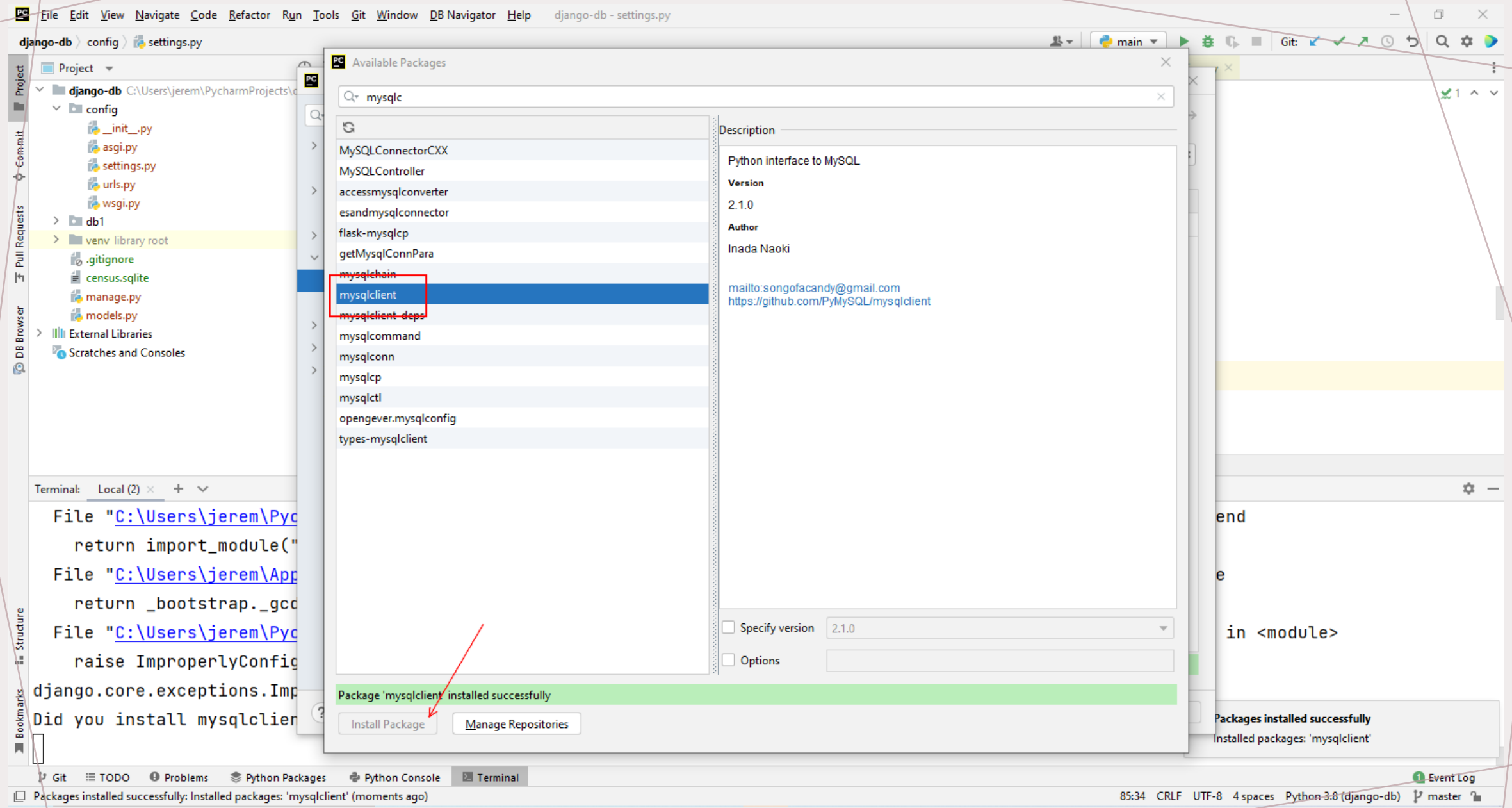

File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
db1
venv library root
.gitignore
census.sqlite
db.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
settings.py
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3',
'NAME': BASE_DIR / 'db.sqlite',
}
}
# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators
Terminal: Local (2)
b1, sessions.
Run 'python manage.py migrate' to apply them.
March 28, 2022 - 16:36:03
Django version 4.0.3, using settings 'config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Git TODO Problems Python Packages Python Console Terminal
Event Log
93:9 CRLF UTF-8 4 spaces Python 3.8 (django-db) master

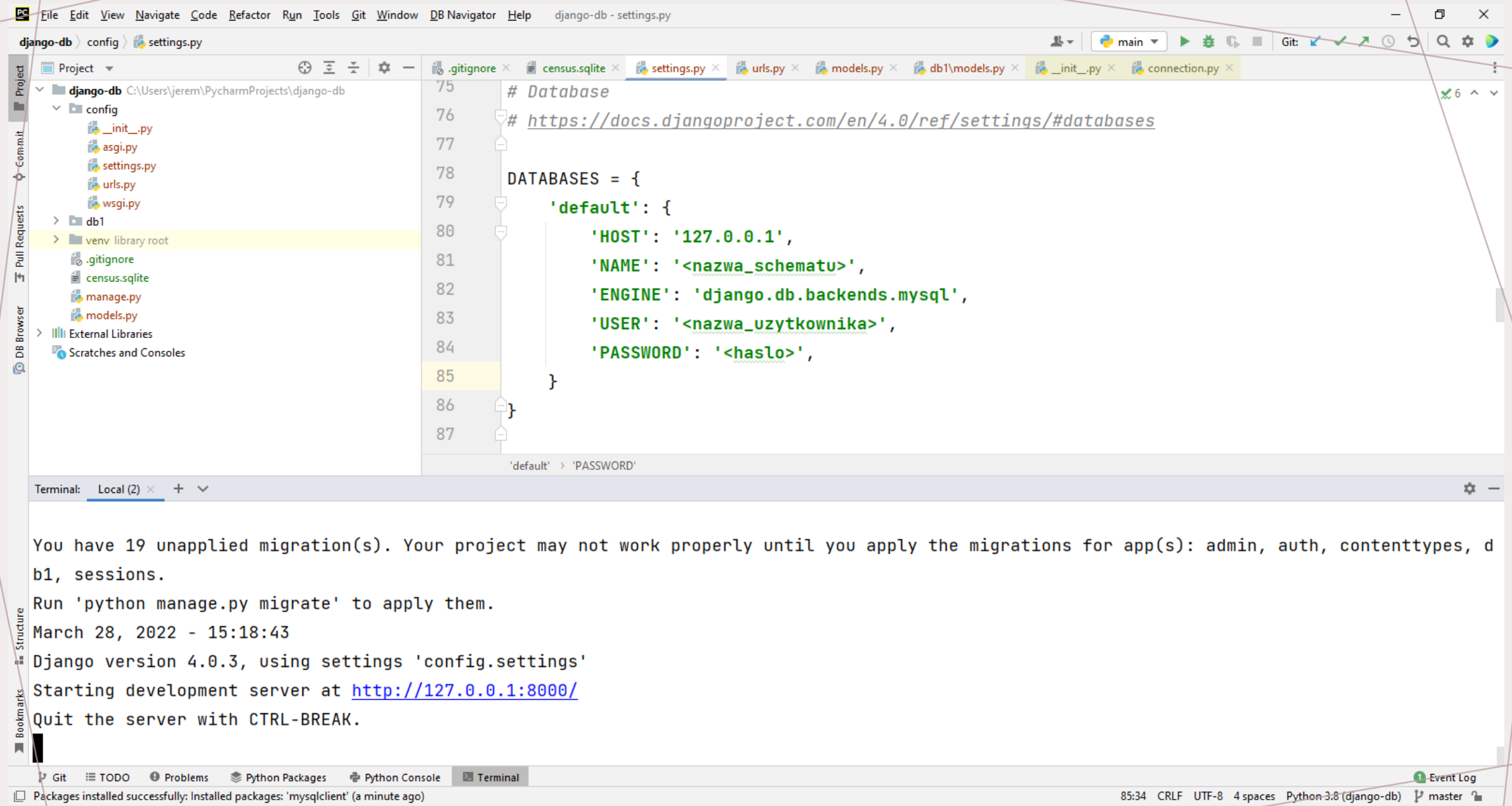

DB Browser for SQLite - C:\Users\jerem\PycharmProjects\django-db\db.sqlite
Plik Edycja Widok Narzędzia Pomoc
Nowa baza danych... Otwórz bazę danych... Zapisz zmiany Wycofaj zmiany Otwórz projekt Zapisz projekt Dołącz bazę danych Zamknij bazę danych
Struktura danych Przeglądarka danych Polecenia Pragmy Polecenia SQL
Utwórz tabelę... Utwórz indeks... Zmień tabelę... Usuń tabelę... Drukuj
Nazwa Rodzaj Polecenie tworzące
Tabele (11)
auth_group CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE)
auth_group_permissions CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group"
auth_permission CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_conter
auth_user CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_sup
auth_user_groups CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEF
auth_user_user_permissions CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user
django_admin_log CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "action_time" datetime NOT NULL, "object_id" text NULL, "objec
django_content_type CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) N
django_migrations CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL,
django_session CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL)
sqlite_sequence CREATE TABLE sqlite_sequence(name,seq)
Indeksy (15)
auth_group_permissions_group_id_b1... CREATE INDEX "auth_group_permissions_group_id_b120cbf9" ON "auth_group_permissions" ("group_id")
auth_group_permissions_group_id_pe... CREATE UNIQUE INDEX "auth_group_permissions_group_id_permission_id_0cd325b0_uniq" ON "auth_group_permissions" ("group_id", "permission_id")
auth_group_permissions_permission_i... CREATE INDEX "auth_group_permissions_permission_id_84c5c92e" ON "auth_group_permissions" ("permission_id")
auth_permission_content_type_id_2f4... CREATE INDEX "auth_permission_content_type_id_2f476e4b" ON "auth_permission" ("content_type_id")
auth_permission_content_type_id_cod... CREATE UNIQUE INDEX "auth_permission_content_type_id_codename_01ab375a_uniq" ON "auth_permission" ("content_type_id", "codename")
auth_user_groups_group_id_97559544 CREATE INDEX "auth_user_groups_group_id_97559544" ON "auth_user_groups" ("group_id")
auth_user_groups_user_id_6a12ed8b CREATE INDEX "auth_user_groups_user_id_6a12ed8b" ON "auth_user_groups" ("user_id")
auth_user_groups_user_id_group_id_... CREATE UNIQUE INDEX "auth_user_groups_user_id_group_id_94350c0c_uniq" ON "auth_user_groups" ("user_id", "group_id")
auth_user_user_permissions_permissi... CREATE INDEX "auth_user_user_permissions_permission_id_1fbb5f2c" ON "auth_user_user_permissions" ("permission_id")
auth_user_user_permissions_user_id_... CREATE INDEX "auth_user_user_permissions_user_id_a95ead1b" ON "auth_user_user_permissions" ("user_id")
auth_user_user_permissions_user_id_... CREATE UNIQUE INDEX "auth_user_user_permissions_user_id_permission_id_14a6b632_uniq" ON "auth_user_user_permissions" ("user_id", "permission_id"
django_admin_log_content_type_id_c... CREATE INDEX "django_admin_log_content_type_id_c4bce8eb" ON "django_admin_log" ("content_type_id")
django_admin_log_user_id_c564eba6 CREATE INDEX "django_admin_log_user_id_c564eba6" ON "django_admin_log" ("user_id")
django_content_type_app_label_mode... CREATE UNIQUE INDEX "django_content_type_app_label_model_76bd3d3b_uniq" ON "django_content_type" ("app_label", "model")
django_session_expire_date_a5c62663 CREATE INDEX "django_session_expire_date_a5c62663" ON "django_session" ("expire_date")
Widoki (0)
Wyzwalacze (0)
Zmiana komórki bazy danych
Tryb: Tekst
1
Rodzaj danych obecnie znajdujących się w komórce
Rozmiar danych znajdujących się obecnie w tabeli
Zastosuj
UTF-8

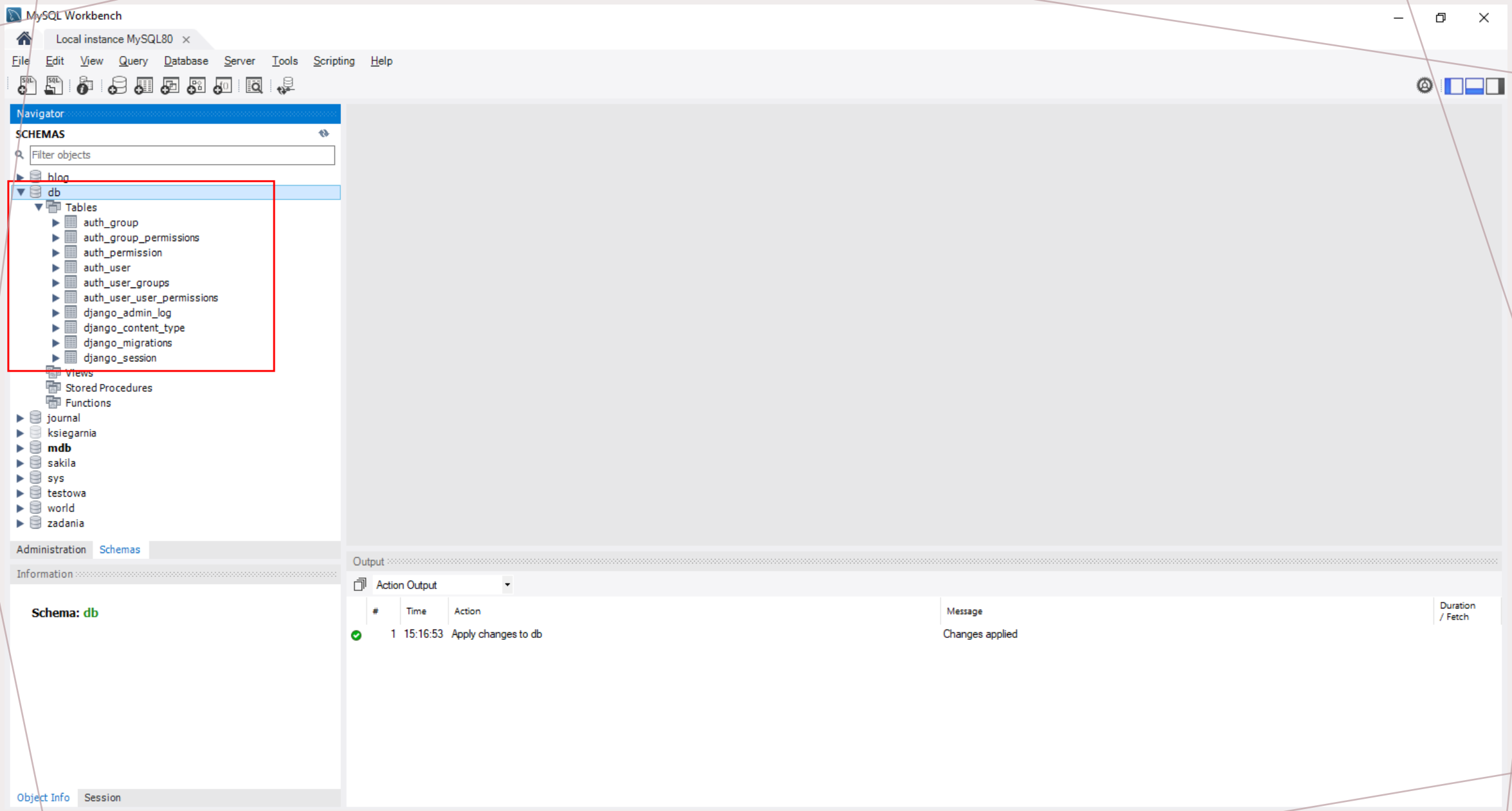
MySQL


File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
'default': {
'HOST': '127.0.0.1',
'NAME': '<nazwa_schematu>',
'ENGINE': 'django.db.backends.mysql',
'USER': '<nazwa_uzytkownika>',
'PASSWORD': '<haslo>',
}
}
'default' › 'PASSWORD'
Terminal: Local (2)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\utils.py", line 113, in load_backend
return import_module("%s.base" % backend_name)
File "C:\Users\jerem\AppData\Local\Programs\Python\Python38\lib\importlib__init__.py", line 127, in import_module
return _bootstrap._gcd_import(name[level:], package, level)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\backends\mysql\base.py", line 17, in <module>
raise ImproperlyConfigured(
django.core.exceptions.ImproperlyConfigured: Error loading MySQLdb module.
Did you install mysqlclient?
Git TODO Problems Python Packages Python Console Terminal
Event Log
Packages installed successfully: Installed packages: 'psycopg2' (16 minutes ago)
85:34 CRLF UTF-8 4 spaces Python 3.8 (django-db) master

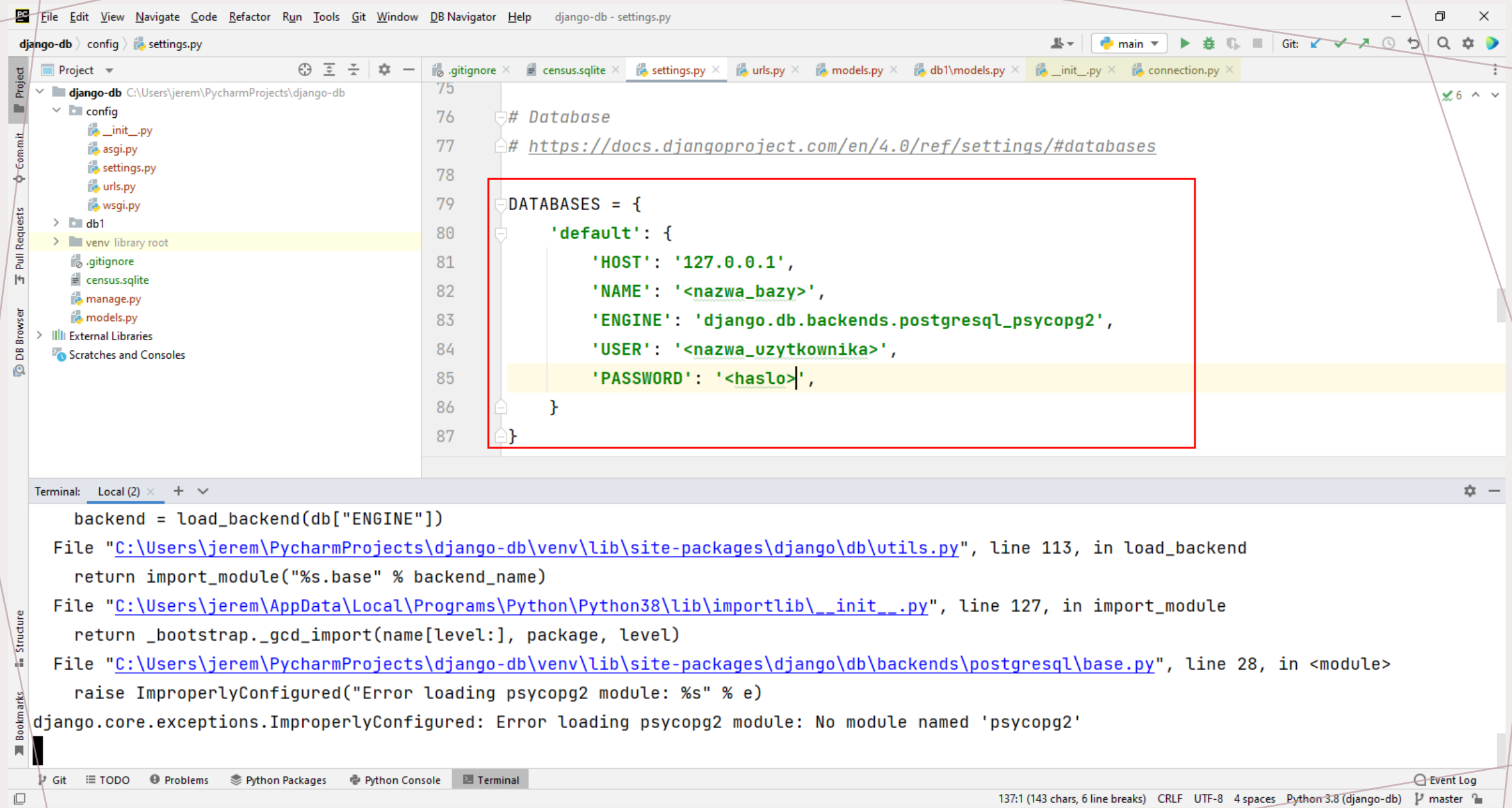

File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
'default': {
'HOST': '127.0.0.1',
'NAME': '<nazwa_schematu>',
'ENGINE': 'django.db.backends.mysql',
'USER': '<nazwa_uzytkownika>',
'PASSWORD': '<haslo>',
}
}
'default' › 'PASSWORD'
Terminal: Local (2)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\utils.py", line 113, in load_backend
return import_module("%s.base" % backend_name)
File "C:\Users\jerem\AppData\Local\Programs\Python\Python38\lib\importlib__init__.py", line 127, in import_module
return _bootstrap._gcd_import(name[level:], package, level)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\backends\mysql\base.py", line 17, in <module>
raise ImproperlyConfigured(
django.core.exceptions.ImproperlyConfigured: Error loading MySQLdb module.
Did you install mysqlclient?
Git TODO Problems Python Packages Python Console Terminal
Event Log
Packages installed successfully: Installed packages: 'psycopg2' (16 minutes ago)
85:34 CRLF UTF-8 4 spaces Python 3.8 (django-db) master

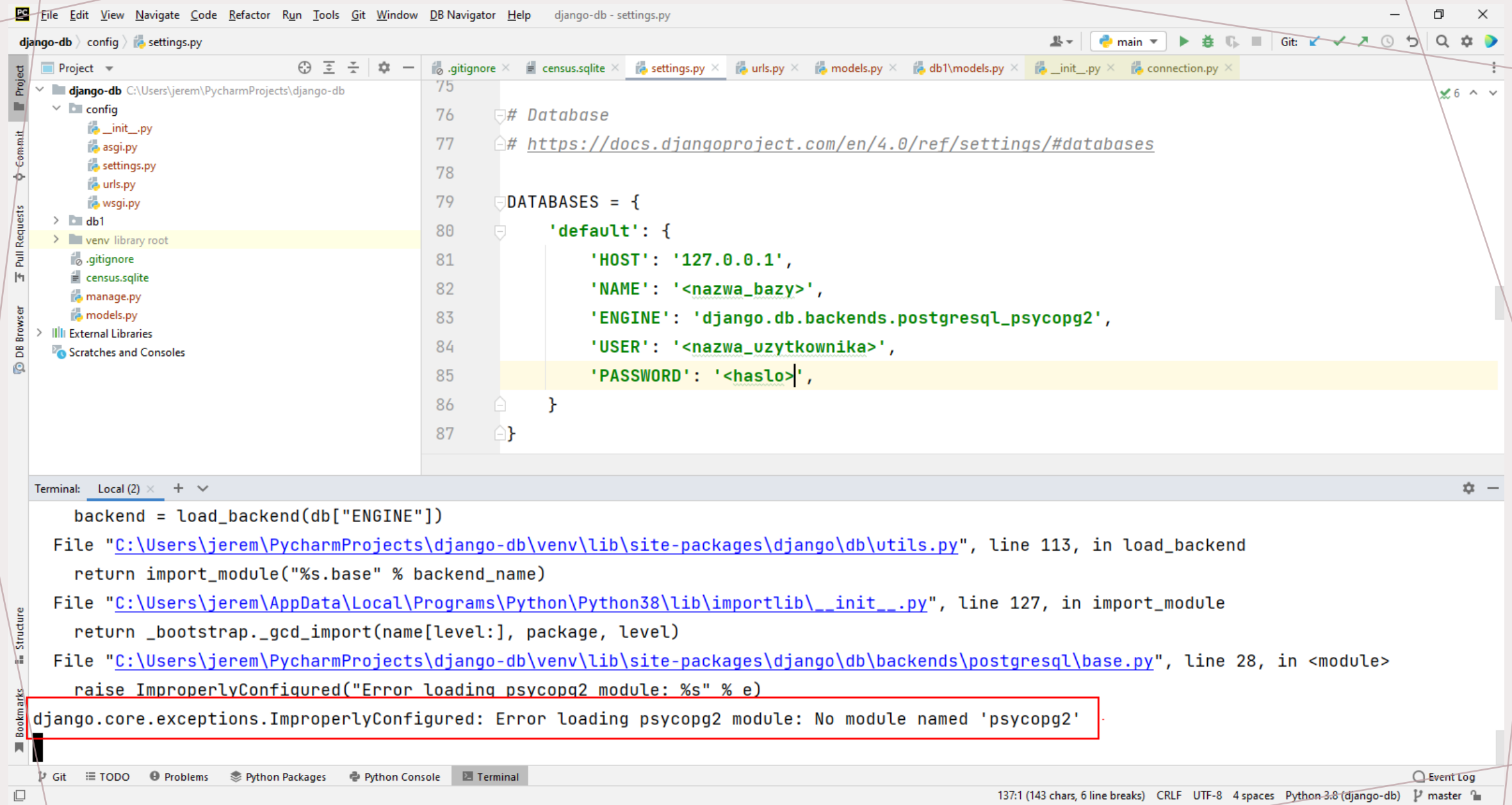

File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
Available Packages
mysqlc
MySQLConnectorCXX
MySQLController
accessmysqlconverter
esandmysqlconnector
flask-mysqlcp
getMysqlConnPara
mysqlchain
mysqlclient
mysqlclient_deps
mysqlcommand
mysqlconn
mysqlcp
mysqlctl
opengever.mysqlconfig
types-mysqlclient
Description
Python interface to MySQL
Version
2.1.0
Author
Inada Naoki
mailto:songofacandy@gmail.com
https://github.com/PyMySQL/mysqlclient
Specify version
2.1.0
Options
Package 'mysqlclient' installed successfully
Install Package
Manage Repositories
Packages installed successfully
Installed packages: 'mysqlclient'
Terminal:
Local (2)
Git
TODO
Problems
Python Packages
Python Console
Terminal
Event Log
Packages installed successfully: Installed packages: 'mysqlclient' (moments ago)
85:34 CRLF UTF-8 4 spaces Python 3.8 (django-db) master

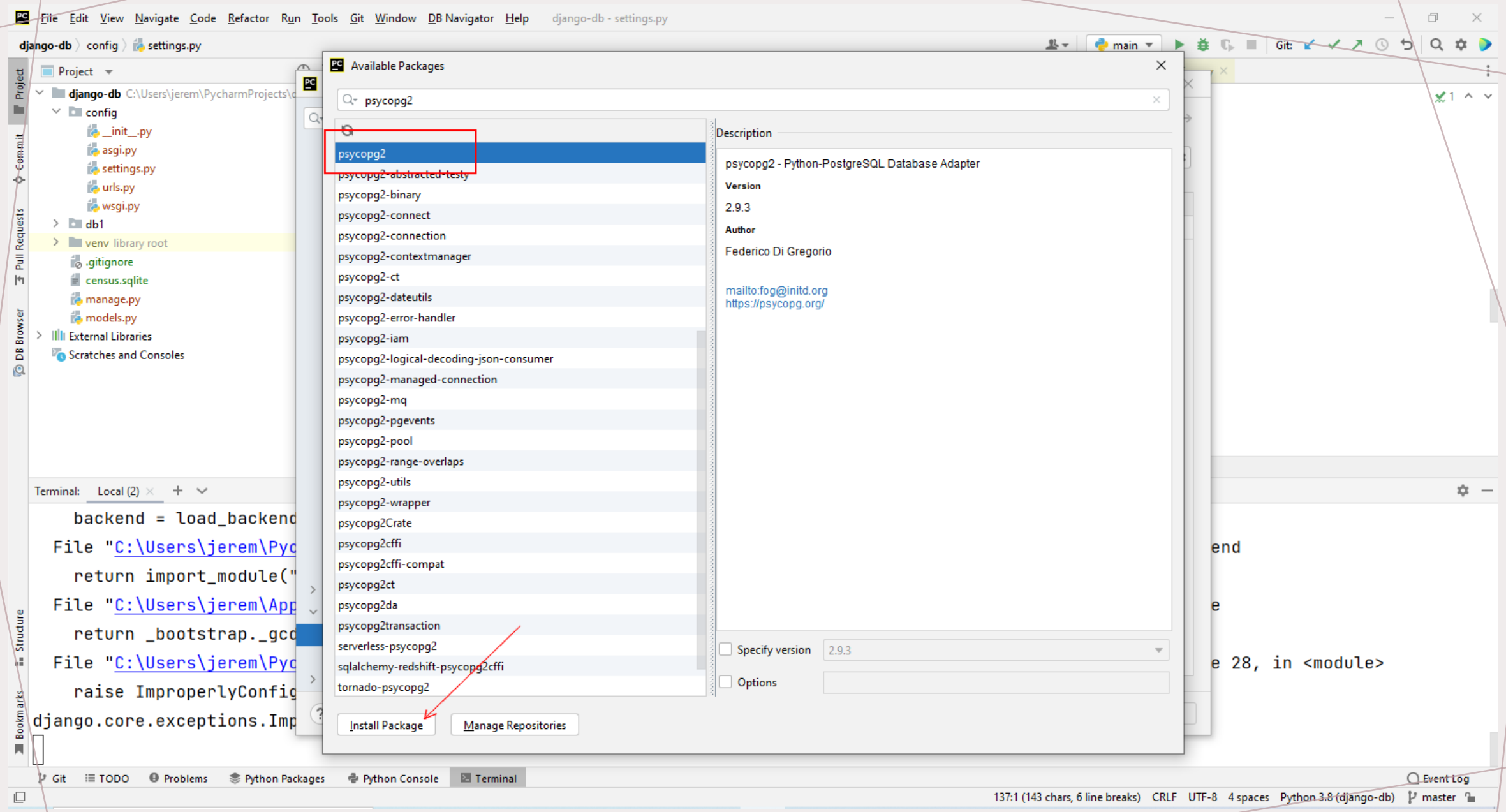

File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases
DATABASES = {
'default': {
'HOST': '127.0.0.1',
'NAME': '<nazwa_schematu>',
'ENGINE': 'django.db.backends.mysql',
'USER': '<nazwa_uzytkownika>',
'PASSWORD': '<haslo>',
}
}
'default' › 'PASSWORD'
Terminal: Local (2)
You have 19 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, db1, sessions.
Run 'python manage.py migrate' to apply them.
March 28, 2022 - 15:18:43
Django version 4.0.3, using settings 'config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Git TODO Problems Python Packages Python Console Terminal
Event Log
Packages installed successfully: Installed packages: 'mysqlclient' (a minute ago)
85:34 CRLF UTF-8 4 spaces Python 3.8 (django-db) master

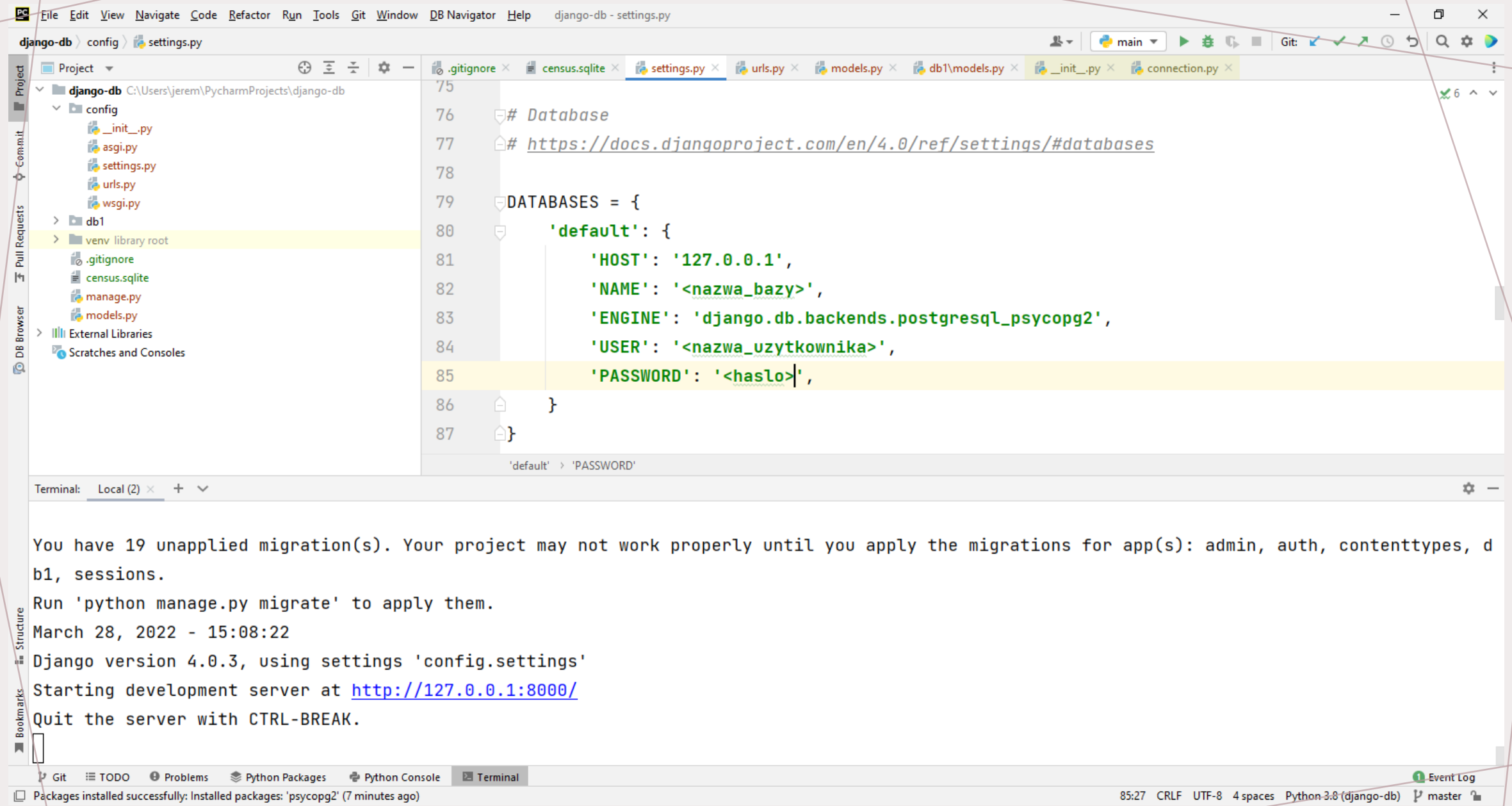

MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
blog
db
Tables
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
Views
Stored Procedures
Functions
journal
ksiegarnia
mdb
sakila
sys
testowa
world
zadania
Administration Schemas
Information
Schema: db
Object Info Session
Output
Action Output
# Time Action Message Duration / Fetch
1 15:16:53 Apply changes to db Changes applied

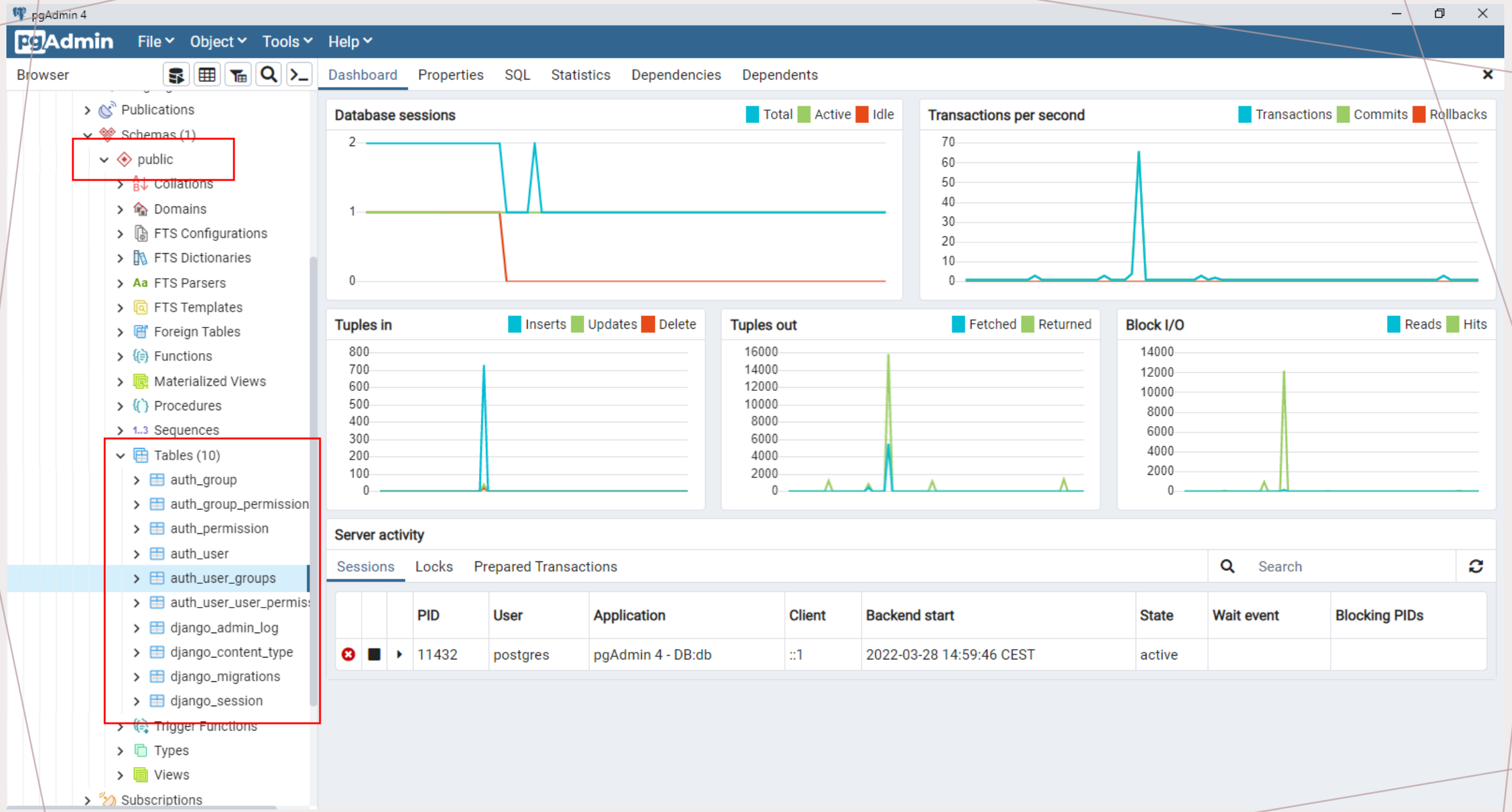
PostgreSQL


File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
75
76 # Database
77 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
78
79 DATABASES = {
80 'default': {
81 'HOST': '127.0.0.1',
82 'NAME': '<nazwa_bazy>',
83 'ENGINE': 'django.db.backends.postgresql_psycopg2',
84 'USER': '<nazwa_uzytkownika>',
85 'PASSWORD': '<haslo>',
86 }
87 }
Terminal: Local (2)
backend = load_backend(db["ENGINE"])
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\utils.py", line 113, in load_backend
return import_module("%s.base" % backend_name)
File "C:\Users\jerem\AppData\Local\Programs\Python\Python38\lib\importlib__init__.py", line 127, in import_module
return _bootstrap._gcd_import(name[level:], package, level)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\backends\postgresql\base.py", line 28, in <module>
raise ImproperlyConfigured("Error loading psycopg2 module: %s" % e)
django.core.exceptions.ImproperlyConfigured: Error loading psycopg2 module: No module named 'psycopg2'
Git TODO Problems Python Packages Python Console Terminal
Event Log
137:1 (143 chars, 6 line breaks) CRLF UTF-8 4 spaces Python 3.8 (django-db) master


File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
75
76 # Database
77 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
78
79 DATABASES = {
80 'default': {
81 'HOST': '127.0.0.1',
82 'NAME': '<nazwa_bazy>',
83 'ENGINE': 'django.db.backends.postgresql_psycopg2',
84 'USER': '<nazwa_uzytkownika>',
85 'PASSWORD': '<haslo>',
86 }
87 }
Terminal: Local (2)
backend = load_backend(db["ENGINE"])
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\utils.py", line 113, in load_backend
return import_module("%s.base" % backend_name)
File "C:\Users\jerem\AppData\Local\Programs\Python\Python38\lib\importlib__init__.py", line 127, in import_module
return _bootstrap._gcd_import(name[level:], package, level)
File "C:\Users\jerem\PycharmProjects\django-db\venv\lib\site-packages\django\db\backends\postgresql\base.py", line 28, in <module>
raise ImproperlyConfigured("Error loading psycopg2 module: %s" % e)
django.core.exceptions.ImproperlyConfigured: Error loading psycopg2 module: No module named 'psycopg2'
Git TODO Problems Python Packages Python Console Terminal
Event Log
137:1 (143 chars, 6 line breaks) CRLF UTF-8 4 spaces Python 3.8 (django-db) master


File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
Available Packages
psycopg2
psycopg2
psycopg2-abstracted-testy
psycopg2-binary
psycopg2-connect
psycopg2-connection
psycopg2-contextmanager
psycopg2-ct
psycopg2-dateutils
psycopg2-error-handler
psycopg2-iam
psycopg2-logical-decoding-json-consumer
psycopg2-managed-connection
psycopg2-mq
psycopg2-pgevents
psycopg2-pool
psycopg2-range-overlaps
psycopg2-utils
psycopg2-wrapper
psycopg2Crate
psycopg2cffi
psycopg2cffi-compat
psycopg2ct
psycopg2da
psycopg2transaction
serverless-psycopg2
sqlalchemy-redshift-psycopg2cffi
tornado-psycopg2
Description
psycopg2 - Python-PostgreSQL Database Adapter
Version
2.9.3
Author
Federico Di Gregorio
mailto:fog@initd.org
https://psycopg.org/
Specify version
2.9.3
Options
Install Package
Manage Repositories
Terminal: Local (2)
backend = load_backend
File "C:\Users\jerem\Pyc
return import_module("
File "C:\Users\jerem\App
return _bootstrap._gcd
File "C:\Users\jerem\Pyc
raise ImproperlyConfig
django.core.exceptions.Imp
Git TODO Problems Python Packages Python Console Terminal
Event Log
137:1 (143 chars, 6 line breaks) CRLF UTF-8 4 spaces Python 3.8 (django-db) master


File Edit View Navigate Code Refactor Run Tools Git Window DB Navigator Help
django-db - settings.py
django-db config settings.py
main
Git:
Project
django-db C:\Users\jerem\PycharmProjects\django-db
config
__init__.py
asgi.py
settings.py
urls.py
wsgi.py
db1
venv library root
.gitignore
census.sqlite
manage.py
models.py
External Libraries
Scratches and Consoles
.gitignore
census.sqlite
settings.py
urls.py
models.py
db1\models.py
__init__.py
connection.py
75
76 # Database
77 # https://docs.djangoproject.com/en/4.0/ref/settings/#databases
78
79 DATABASES = {
80 'default': {
81 'HOST': '127.0.0.1',
82 'NAME': '<nazwa_bazy>',
83 'ENGINE': 'django.db.backends.postgresql_psycopg2',
84 'USER': '<nazwa_uzytkownika>',
85 'PASSWORD': '<haslo>',
86 }
87 }
'default' › 'PASSWORD'
Terminal: Local (2)
You have 19 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, db1, sessions.
Run 'python manage.py migrate' to apply them.
March 28, 2022 - 15:08:22
Django version 4.0.3, using settings 'config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
Git TODO Problems Python Packages Python Console Terminal
Event Log
Packages installed successfully: Installed packages: 'psycopg2' (7 minutes ago)
85:27 CRLF UTF-8 4 spaces Python 3.8 (django-db) master


pgAdmin 4
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Publications
Schemas (1)
public
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
Tables (10)
auth_group
auth_group_permission
auth_permission
auth_user
auth_user_groups
auth_user_user_permis
django_admin_log
django_content_type
django_migrations
django_session
Trigger Functions
Types
Views
Subscriptions
Database sessions
Total
Active
Idle
Transactions per second
Transactions
Commits
Rollbacks
Tuples in
Inserts
Updates
Delete
Tuples out
Fetched
Returned
Block I/O
Reads
Hits
Server activity
Sessions
Locks
Prepared Transactions
Search
PID
User
Application
Client
Backend start
State
Wait event
Blocking PIDs
11432
postgres
pgAdmin 4 - DB:db
::1
2022-03-28 14:59:46 CEST
active

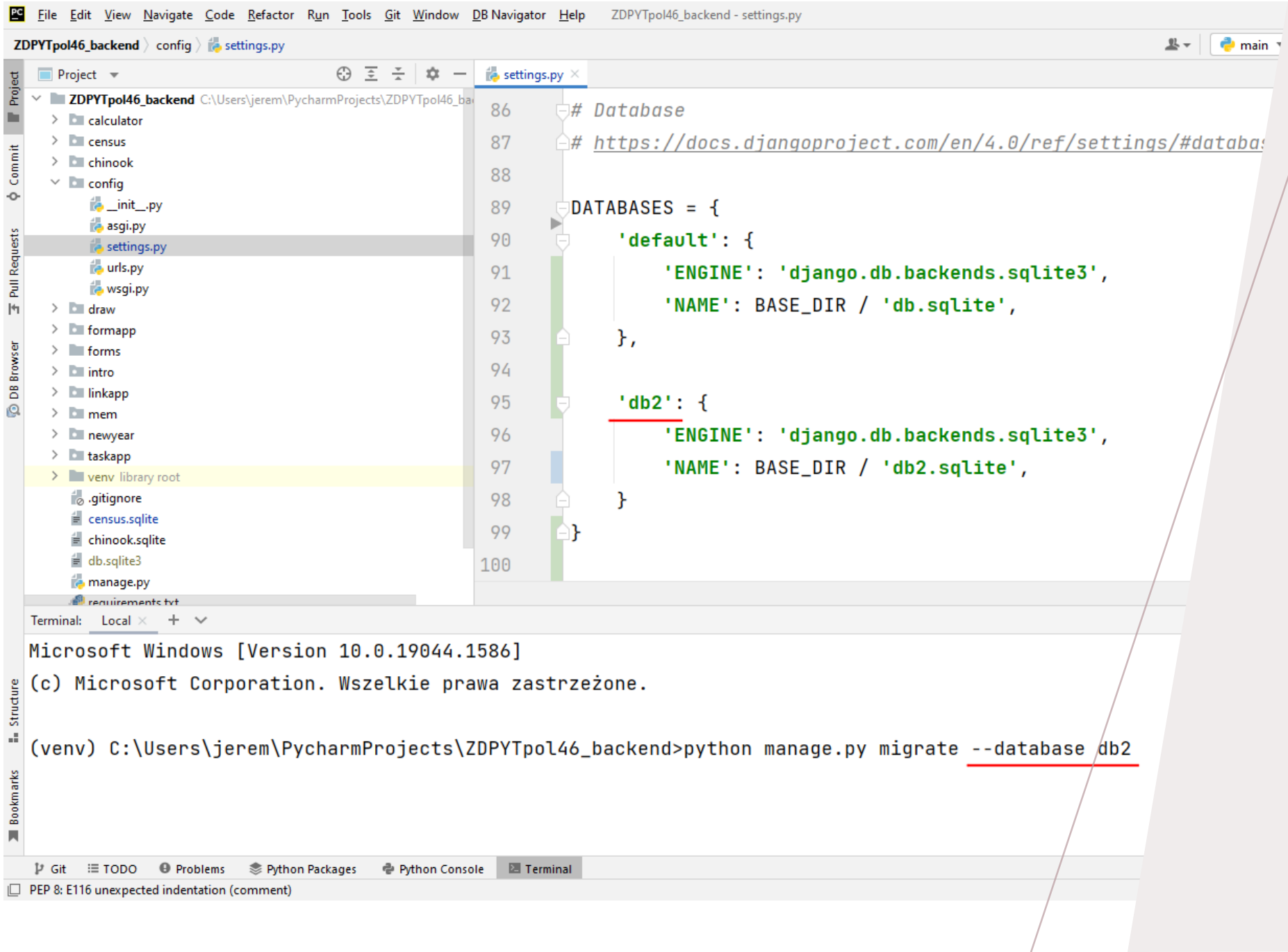
# *WSPARCIE DLA WIELU BAZ*

# *WIELE BAZ W JEDNYM PROJEKCIE*

Django umożliwia skonfigurowanie wielu baz w jednym projekcie. W ramach jednego projektu można mieć podpięte różne typy baz. W takim przypadku przy wykonywaniu migracji należy wskazać "nazwę bazy" (klucz w zmiennej DATABASES w pliku settings.py), na której chcemy wykonać migrację. Do wskazywania "nazwy bazy" służy parametr  --database komendy python manage.py migrate.

*python manage.py migrate - -database <nazwa_bazy>*

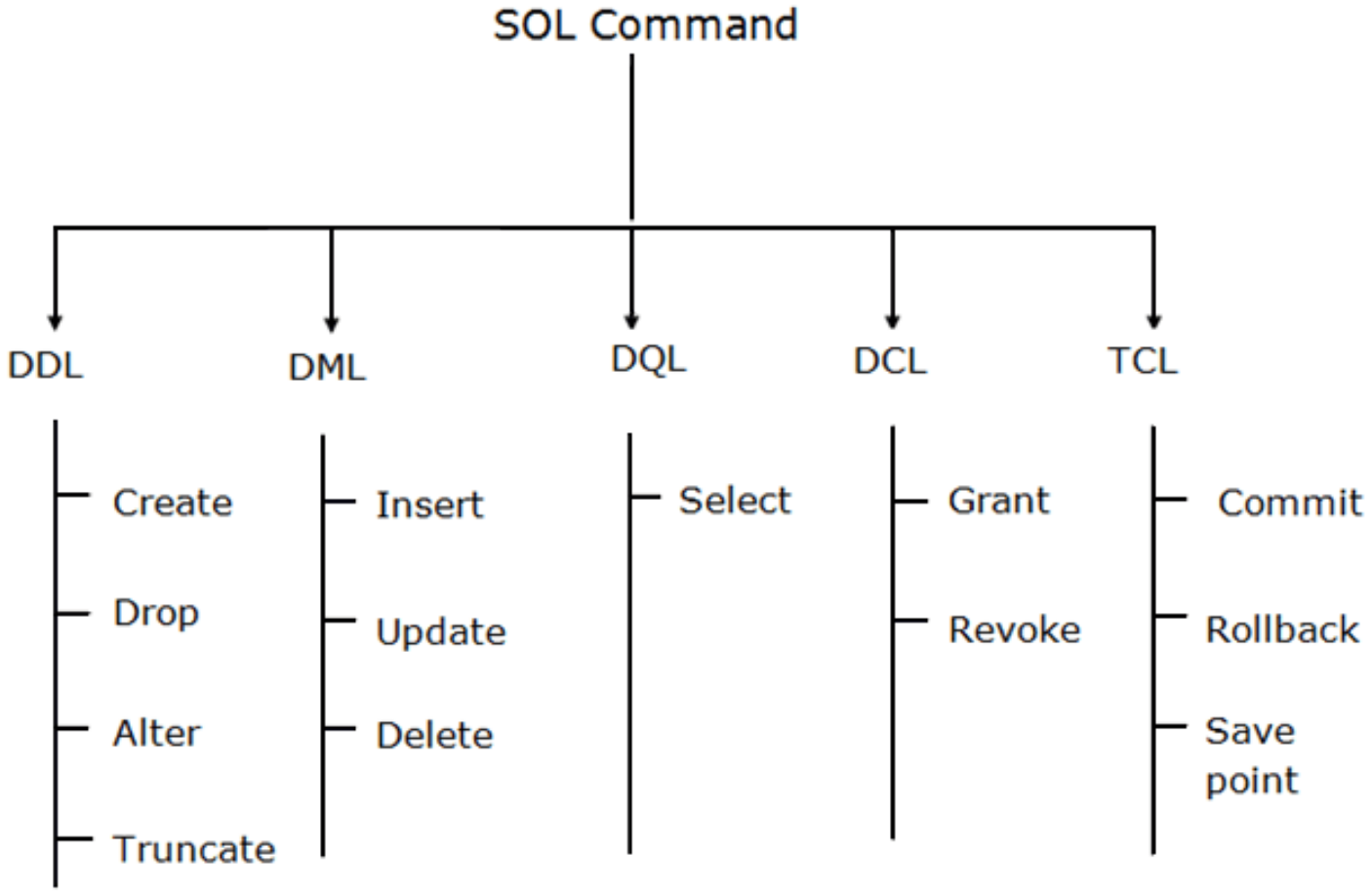
Domyślnie, przy braku parametru  - -database, django szuka konfiguracji dla nazwy 'default'.




# MULTI DATABASE SUPPORT

# PODSTAWOWE INSTRUKCJE SQL

- **DDL** – Data Definition Language
- **DML** – Data Manipulation Language
- **DQL** – Data Query Language
- **DCL** – Data Control Language
- **TCL** – Transaction Control Language



SERWER

KLIENT

# ARCHITEKTURA KLIENT-SERWER