***Django 5***

***Dawid Jędraszczyk***

***148293***

0. Słowem wstępu

Jeżeli napotkasz jakieś problemy, podczas tworzenia projektu. Pobierz go i analizuj kod. Na końcu tego tutoriala, jest link do GitHuba.

1. Czym jest Django?

**Django** to wysokopoziomowy framework webowy napisany w języku Python, który ułatwia szybkie tworzenie bezpiecznych i skalowalnych aplikacji internetowych. Django jest znane z tego, że podąża za filozofią "Batteries included", co oznacza, że zawiera wiele gotowych do użycia komponentów, które przyspieszają proces tworzenia aplikacji.

1.1 Cechy charakterystyczne

**Szybkość i wydajność**:

Django pozwala na szybkie tworzenie aplikacji webowych dzięki prostocie rozwoju oprogramowania i komponentom, takim jak generowanie szkieletów kodu, migracje bazy danych i testowanie.

**Bezpieczeństwo**:

Django zawiera wiele wbudowanych mechanizmów bezpieczeństwa, które chronią aplikacje przed najczęstszymi zagrożeniami, takimi jak ataki SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF) i inne.

**Skalowalność**:

Dzięki swojej modularnej budowie, Django łatwo dostosowuje się do rosnących potrzeb aplikacji, pozwalając na efektywne zarządzanie zarówno małymi, jak i dużymi projektami.

**Reużywalność kodu**:

Dzięki zastosowaniu aplikacji i modułów, które można łatwo integrować i ponownie wykorzystywać w różnych projektach, Django promuje dobre praktyki programistyczne i umożliwia efektywne zarządzanie kodem.

1.2 Zastosowanie

Django jest używane przez wiele znanych firm i organizacji do tworzenia różnorodnych aplikacji webowych, w tym serwisów społecznościowych, platform e-commerce, systemów zarządzania treścią (CMS) oraz aplikacji do analizy danych.

2. Instalacja Django

2.1 Musimy rozpocząć od utworzenia wirtualnego środowiska. W miejscu, w którym chcemy aby znajdował się nasz projekt wpisujemy:

***python -m venv venv***

następnie musimy aktywować środowisko wirtualne, zależnie od systemu

win: ***venv/Scripts/activate***

linux/macOS: ***source venv/bin/activate***

W momencie gdy nasze środowisko wirtualne jest już odpalone, możemy zająć się instlacją Django.

***pip install django***

Możemy teraz sprawdzić czy poprawnie zainstalowaliśmy bibliotekę. Sprawdź wersję Django:

***django-admin --version***

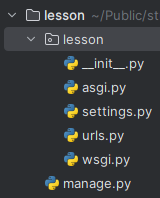
2.2. Tworzenie projektu

Aby utworzyć projekt musimy wpisać komendę

***django-admin startproject NAZWA\_PROJEKTU***

W tym momencie powstał nam projekt w folderze o nazwie projektu. Pierwszy krok za nami :)

Oto struktura plików, która powstała.

* **\_\_init\_\_.py**: Pusty plik, który informuje Pythona, że katalog ten powinien być traktowany jako moduł.
* ****asgi.py****: Plik do konfiguracji serwera ASGI
* ****settings.py****: Główny plik konfiguracyjny projektu. Zawiera wszystkie ustawienia projektu, takie jak konfiguracja bazy danych, aplikacje zainstalowane w projekcie, ścieżki do plików statycznych, szablonów i inne.
* ****urls.py****: Plik do zarządzania mapowaniem URL-i na widoki (views) w projekcie. Określa, które widoki są wywoływane w odpowiedzi na konkretne żądania URL.
* ****wsgi.py****: Plik do konfiguracji serwera WSGI
* **manage.py**: Narzędzie wiersza poleceń Django do zarządzania projektem. Służy do uruchamiania serwera deweloperskiego, wykonywania migracji bazy danych, tworzenia nowych aplikacji i innych zadań administracyjnych.

3. Czym jest aplikacja

**Aplikacja** w Django to niezależny komponent, który wykonuje określone zadania w ramach projektu. Aplikacja może być ponownie używana w różnych projektach. Każda aplikacja ma własne modele, widoki, szablony i pliki statyczne.

**Musimy pamiętać, że projekt jest jeden (w moim przypadku *lesson*), natomiast aplikacji może być wiele.**

Dla przykładu, jeżeli tworzymy aplikację do nauki języków obcych:

* **projektem** jest cały nasz projekt, czyli zbiór wszystkich aplikacji
* **aplikacją** mogą być różne funkcjonalności, np.:

\*\* aplikacja umożliwiająca logowanie/rejestrację,

\*\* aplikacja umożliwiająca zakup pakietu premium,

\*\* aplikacja do nauki słówek.

Czyli tak prosto mówiąc, aplikacja w pewnym sensie odpowiada za fragment projektu. Jego skala jest zależna od twoich decyzji biznesowych. Pamiętaj, że im mniejsze pakiety tych aplikacji, tym łatwiejsze wdrażanie ich w nowych projektach, co pozwala zaoszczędzić wiele czasu.

3.1 Tworzenia aplikacji w Django

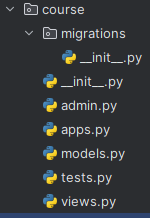
W moim przypadku stworzę aplikację *course* która pozwoli mi na tworzenie kursów.

***python manage.py startapp NAZWA\_APLIKACJI***

w moim przypadku będzie to:

***python manage.py startapp course***

Oto utworzona struktura plików:

* **migrations/**: Katalog zawierający pliki migracji bazy danych, które śledzą zmiany w modelach aplikacji.
* \_\_init\_\_.py: Pusty plik, który informuje Pythona, że katalog ten powinien być traktowany jako moduł.
* **admin.py**: Plik konfiguracyjny do rejestrowania modeli w panelu administracyjnym Django.
* **apps.py**: Plik konfiguracyjny aplikacji, zawierający ustawienia specyficzne dla aplikacji.
* **models.py**: Plik do definiowania modeli aplikacji, które reprezentują dane przechowywane w bazie danych.
* **tests.py**: Plik do definiowania testów jednostkowych dla aplikacji.
* **views.py**: Plik do definiowania widoków, czyli logiki biznesowej aplikacji, która przetwarza żądania i zwraca odpowiedzi.

**3.2 Różnice między projektem a aplikacją**

**Projekt**:

* Zawiera globalne ustawienia i konfiguracje.
* Może składać się z jednej lub więcej aplikacji.
* Zarządza routowaniem URL-i na poziomie globalnym.
* Zawiera pliki konfiguracyjne takie jak settings.py, urls.py, wsgi.py, asgi.py

**Aplikacja**

* Odpowiada za konkretną funkcjonalność w ramach projektu.
* Może być ponownie używana w różnych projektach.
* Zawiera własne modele, widoki, szablony i pliki statyczne.
* Każda aplikacja ma swoje własne pliki takie jak models.py, views.py, admin.py, migrations/.

**3.3 Po stworzeniu aplikacji**

Bardzo ważnym jest aby po stworzeniu aplikacji, dopisać ją na listę INSTALLED\_APPS w settings.py w module naszego projektu.

Tak powinny wyglądać nasze INSTALLED\_APPS

INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'course'  
]

Bez dopisania aplikacji, Django nie będzie sprawdzało w niej zmian i nie będzie pozwalało nam korzystać z niej.

4. Modele

Modele w Django to klasy Pythona, które reprezentują struktury danych w bazie danych. Każdy model odpowiada jednej tabeli w bazie danych (encji), a każdy atrybut modelu odpowiada kolumnie w tej tabeli. Django dostarcza łatwego sposobu definiowania modeli oraz automatycznego tworzenia i zarządzania bazą danych za pomocą migracji.

Przykładowy model reprezentujący kurs w aplikacji do nauki:

from django.db import models

class Course(models.Model):

title = models.CharField(max\_length=200) # Kolumna tekstowa z ograniczoną liczbą znaków

content = models.TextField() # Kolumna tekstowa bez ograniczenia długości

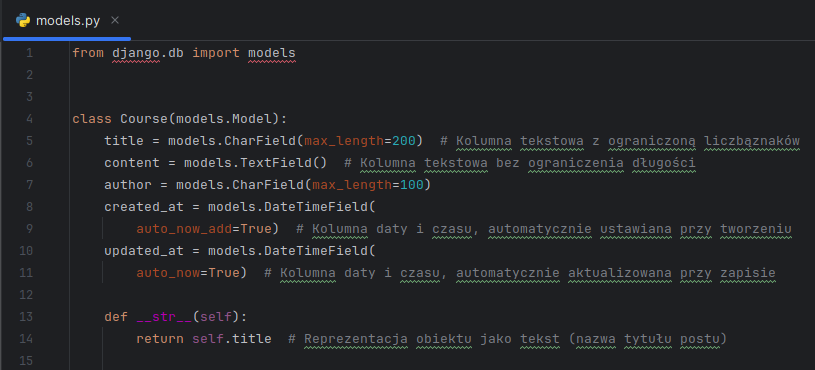
author = models.CharField(max\_length=100)

created\_at = models.DateTimeField(auto\_now\_add=True) # Kolumna daty i czasu, automatycznie ustawiana przy tworzeniu

updated\_at = models.DateTimeField(auto\_now=True) # Kolumna daty i czasu, automatycznie aktualizowana przy zapisie

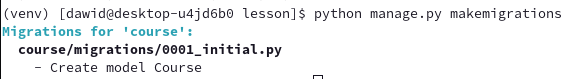
def \_\_str\_\_(self):

return self.title # Reprezentacja obiektu jako tekst (nazwa tytułu postu)

  
Teraz gdy mamy już mamy przepisany nasz kod do models.py w aplikacji Course. Musimy wykonać dwie komendy

***python manage.py makemigrations***

Komenda makemigrations przegląda nasz projekt w celu znalezienia zmian w modelach. Gdy takowe znajdzie tworzy plik migracji, który te zmiany opisuje.



W tym przypadku, zmianą było stworzenie modelu Course. Dokładny opis migracji znajduje się w course/migrations/0001\_initial.py

Aby nasza baza danych uwzględniała zmiany modeli musimy wykonać komendę:

***python manage.py migrate***

4.1 Tworzenie pierwszego obiektu

Abyśmy mogli stworzyć pierwszy obiekt musimy wykonać komendę

***python manage.py shell***

Przeniesie nas ona do terminala Django, w którym możemy wykonywać wiele operacji, wywoływać funkcje, tworzyć obiekty, itd…

W terminalu musimy wpisać:

***from django.apps import apps***

***apps.get\_model(‘course.Course’).objects.create(***

***title=’TWÓJ TYTUŁ’,***

***content=’TWÓJ KONTENT’,***

***author=’TWOJE IMIE I NAZWISKO’)***

Aby wyjść wciskamy CTRL+C



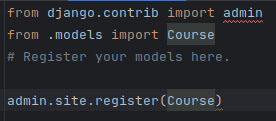
W ten sposób utworzyliśmy pierwszy model django.

4.2 Dostęp do Admina

Abyśmy mieli łatwiejszy dostęp do naszych modeli, możemy je zarejestrować w panelu Admina. W tym celu musimy dopisać kilka linii w pliku: course/admin.py

***from .models import Course***

***admin.site.register(Course)***

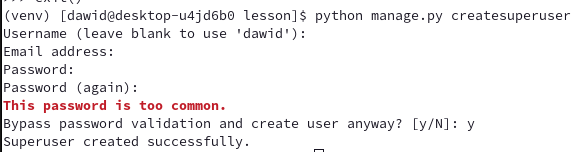


W ten sposób zarejestrowaliśmy możliwość tworzenia, usuwania i przeglądania obiektów Course.

Abyśmy mogli wejść do panelu admina musimy utworzyć konto admina:

***python manage.py createsuperuser***

Następnie wypełniamy danymi

  
W tym momencie nastaje wielka chwila i pierwszy raz odpalimy naszą aplikację!

***Python manage.py runserver***

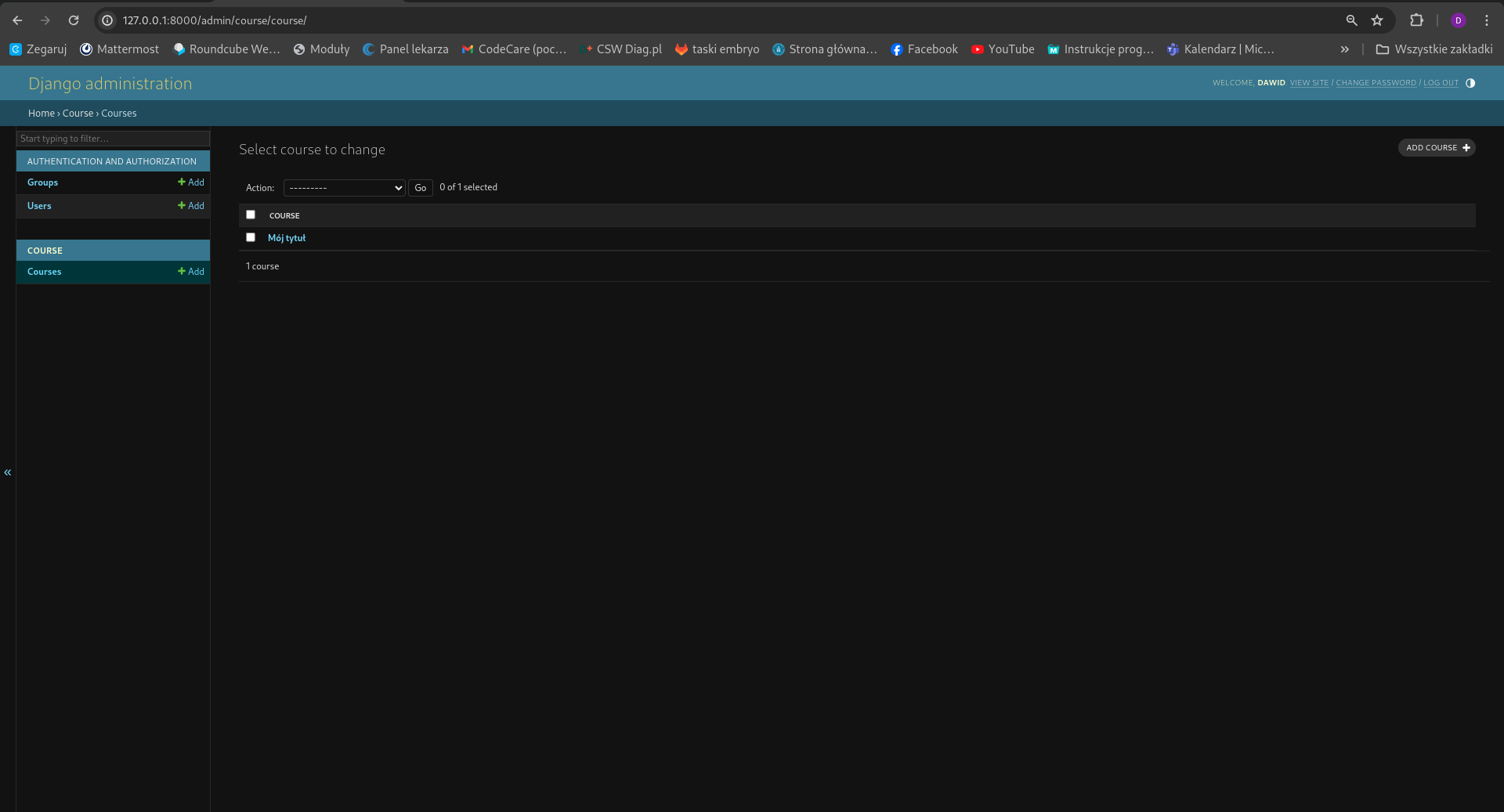
następnie przechodzimy pod link, który pokazany jest w terminalu + /admin

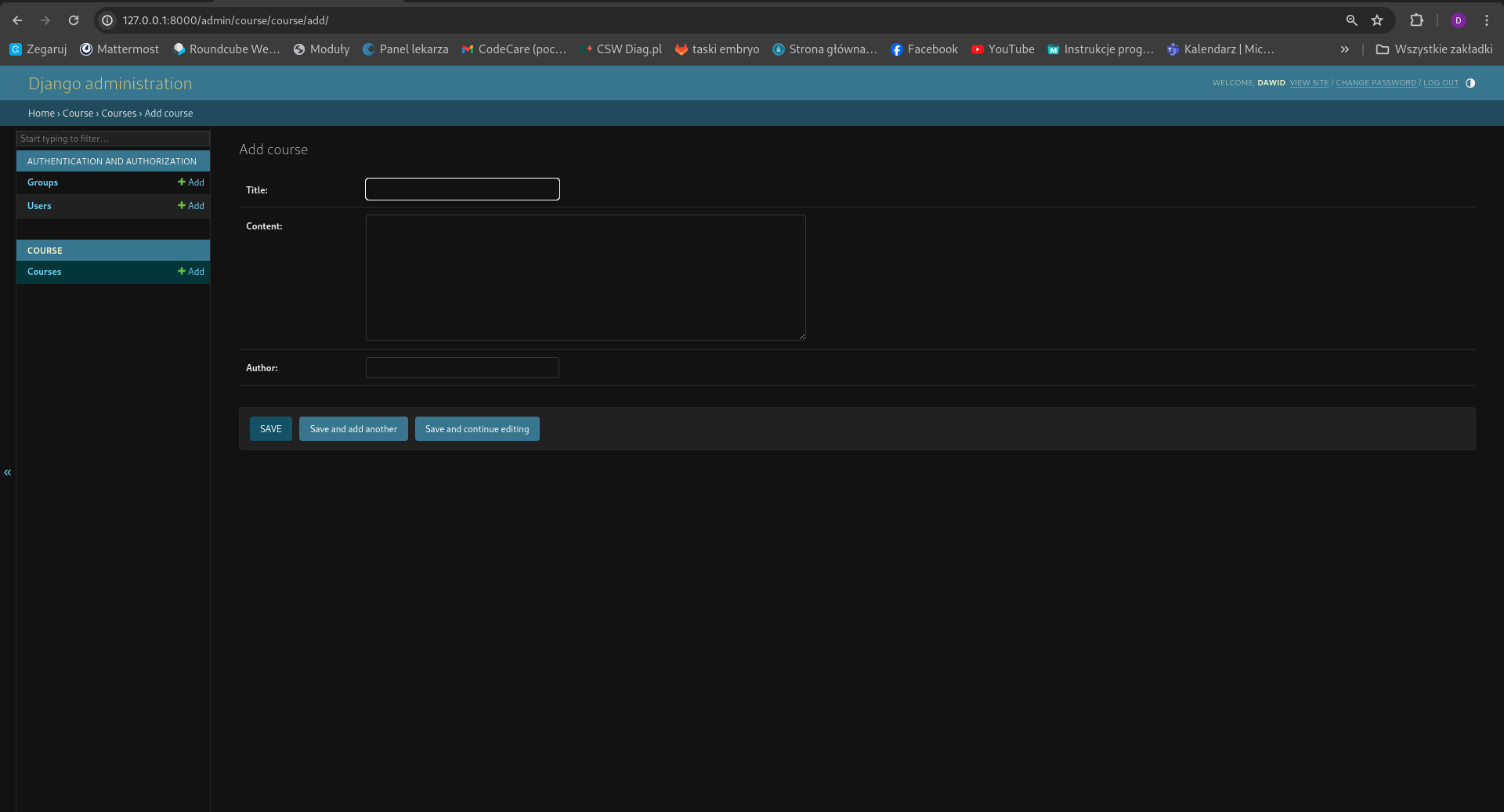
domyślnie: http://127.0.0.1:8000/admin

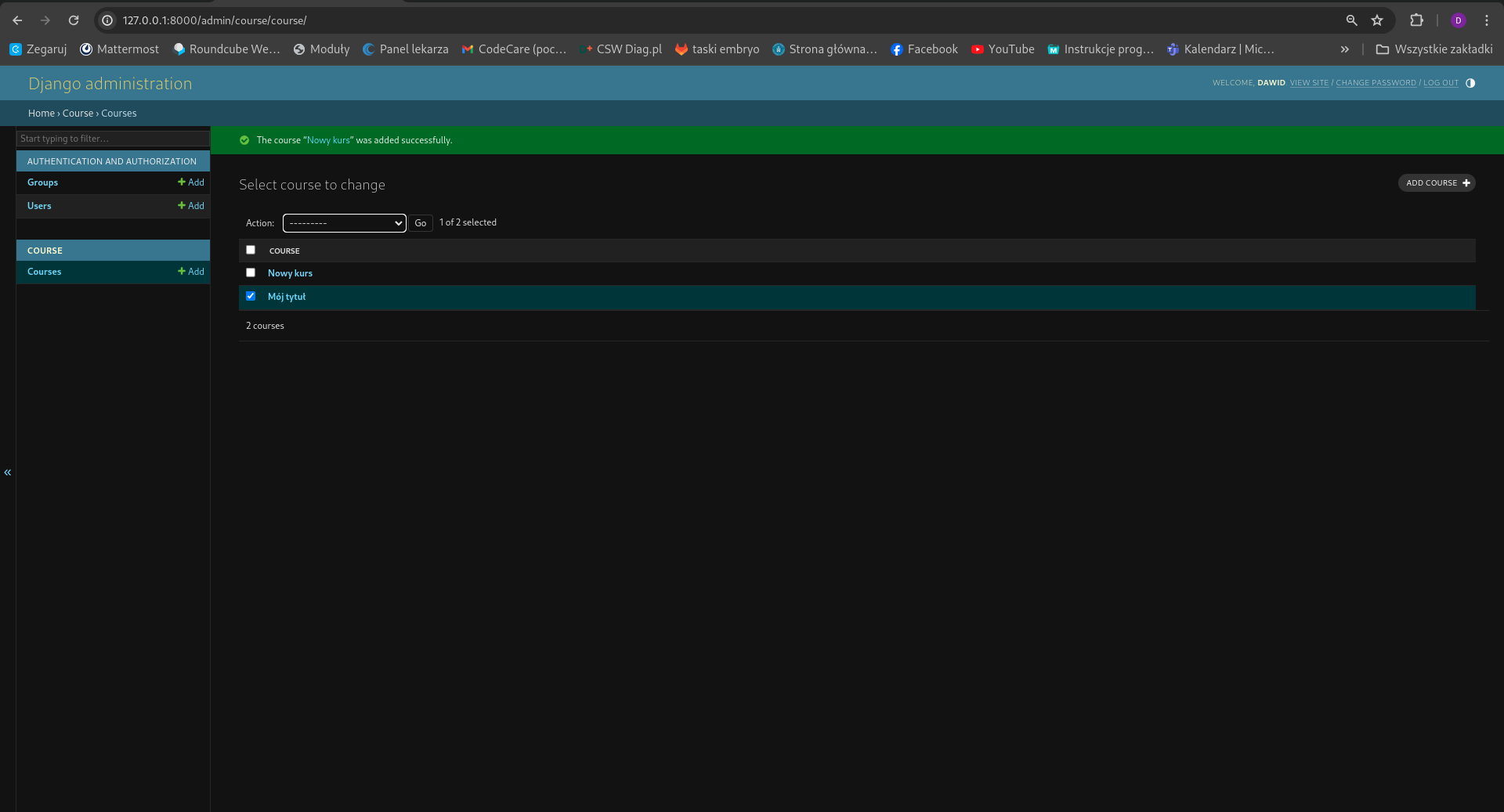
wpisujemy dane użytkownika utworzonego powyżej.

W tym miejscu mamy możliwość wejścia w Courses.

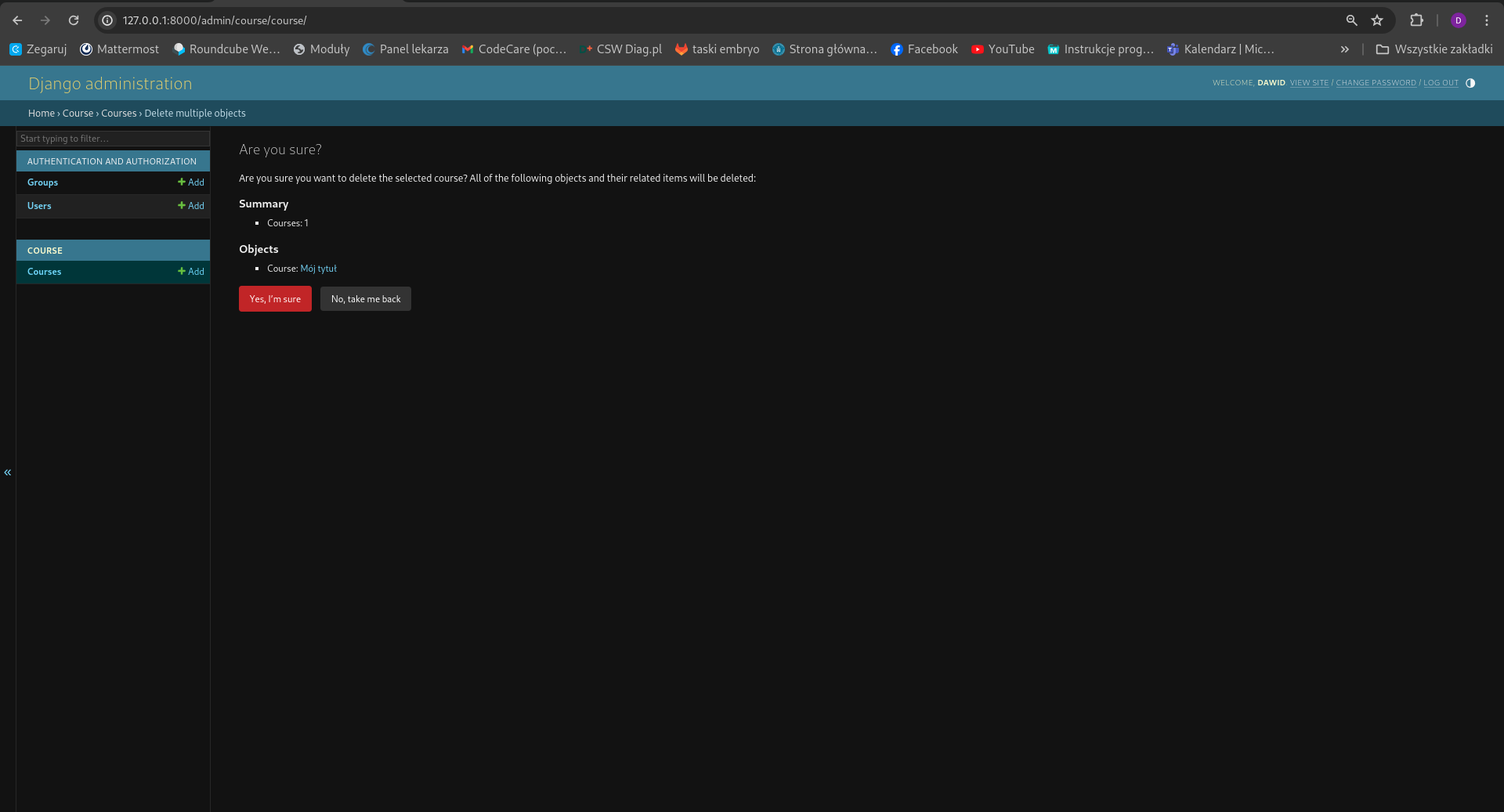
Na ekranie powinno pokazać się coś takiego:

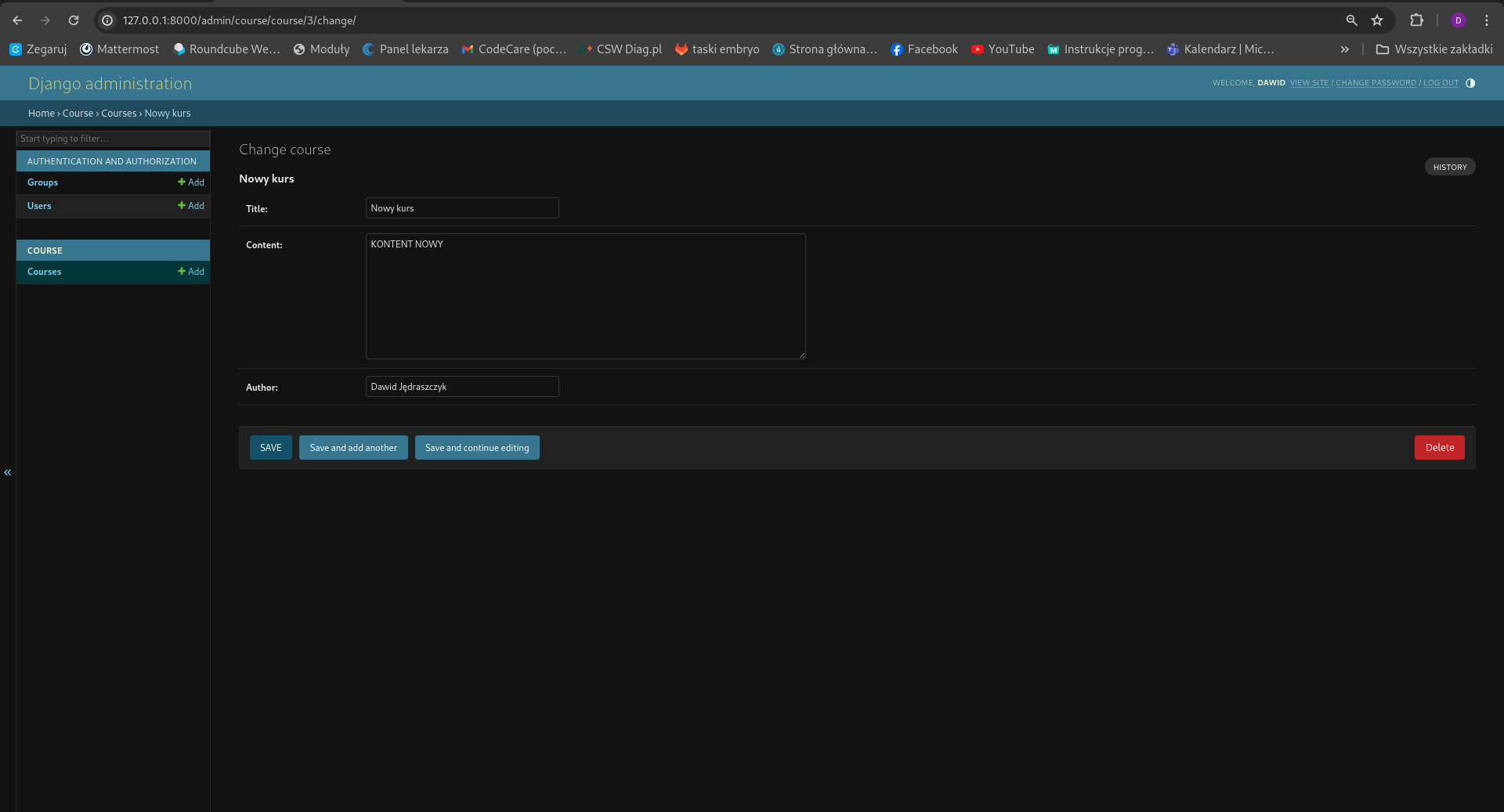
  
Możemy kliknąć w prawym górnym rogu i dodać nowy kurs:

Aby usunąć kurs musimy zaznaczyć kurs który nas interesuje:

  
Nastepnie kliknąć w kreski obok action i wybrać *Delete selected courses*

Potwierdzamy wybór i usunęliśmy wcześniej utworzony kurs.

  
Klikając w nazwę takiego kursu, możemy również wejść w jego szczegóły.



5 Widoki i templates

Widoki są mechanizmem obsługującym logikę biznesową naszej aplikacji. Przetwarzają żądania i zwracają odpowiedzi.

Ważnym elementem widoków jest routing url, mapuje on adresy URL na widoki w naszej aplikacji. Prościej mówiąc, mówimy naszemu serwerowi który widok ma odesłać na żadanie jakiego urla.

Trzecim i równie istotnym elementem co Widoki i Modele są Templates, które są do serwowania użytkownikowi kodu HTML. W django nie musimy pisać czystego HTML’a. Możemy w nim używać pętli, zmiennych i wielu innych rzeczy.

5.1 Stwórzmy pierwszy widok, odpowiadający nam za wyświetlanie listy Kursów

w courses/views.py wpisz kod:

***from django.shortcuts import render***

***from .models import Course***

***def course\_list(request):***

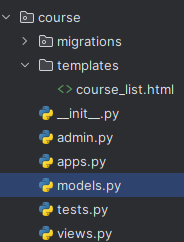
***courses = Course.objects.all()***

***return render(request, ‘course\_list.html’, {‘courses’: courses})***

w powyższym kodzie, stworzyliśmy widok *course\_list,* który pobiera listę wszystkich kursów, następnie zwraca template ‘course\_list.html’, ze zmienną ‘courses’, w której mamy listę naszych kursów.

Następnie musimy stworzyć template dla naszego widoku

w course musimy stworzyć folder templates, następnie w nim plik: ‘course\_list.html’.



Następnie do pliku course\_list wstawiamy ten kod:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Lista Kursów</title>**

**</head>**

**<body>**

**<h1>Lista Kursów</h1>**

**<ul>**

**{% for course in courses %}**

**<li>**

**<h2>{{ course.title }}</h2>**

**<p>{{ course.content }}</p>**

**<p><strong>Autor:</strong> {{ course.author }}</p>**

**<p><em>Utworzono: {{ course.created\_at }}</em></p>**

**<p><em>Aktualizowano: {{ course.updated\_at }}</em></p>**

**</li>**

**{% endfor %}**

**</ul>**

**</body>**

**</html>**

Następnie gdy mamy już nasz widok, wraz z templatem musimy zapewnić URL dla tego widoku.

W courses tworzymy plik urls.py

i wklejamy ten kod:

**from django.urls import path**

**from . import views**

**urlpatterns = [**

**path('', views.course\_list, name='course\_list'),**

**]**

Ostatnie co zostało nam do zrobienia wyjątkowo, jest dodanie URL dla naszej aplikacji do projektu, ponieważ jeszcze tego nie zrobiliśmy.

W pliku lesson/urls.py zmieniamy kod na:

**from django.contrib import admin**

**from django.urls import path, include**

**urlpatterns = [**

**path('admin/', admin.site.urls),**

**path('courses/', include('course.urls')),**

**]**

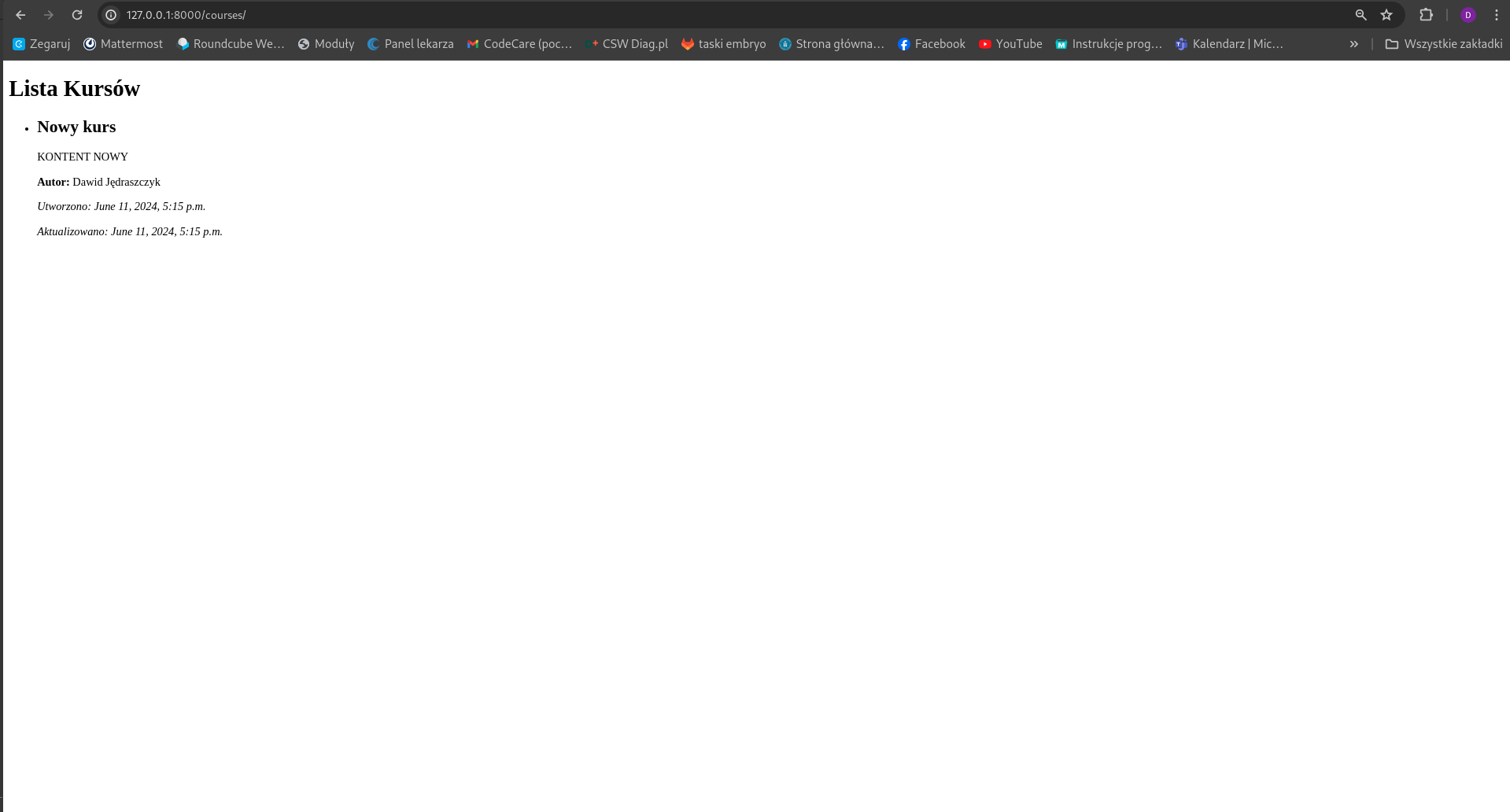
Po uruchomieniu serwera komendą:

***python manage.py runserver***

przechodzimy pod adres url:

127.0.0.1:8000/courses/

Jeżeli wykonaliśmy wszystko poprawnie powinien pokazać nam się taki widok:

  
Gratulację. Stworzyliśmy pierwszy widok, w dodatku związany z modelem :)

6. Formularze

Jest to już ostatni element, który dziś poznamy ;)

Abyśmy mogli tworzyć nowe obiekty, musimy stworzyć formularz. Jest to dosyć proste.

Przykład formularza dla Course. Ten kod wrzucamy do course/forms.py

**from django import forms**

**from .models import Course**

**class CourseForm(forms.ModelForm):**

**class Meta: model = Course**

**fields = ['title', 'content', 'author']**

Następnie dodajemy kod do course/views.py

**from django.shortcuts import render, redirect**

**from .models import Course**

**from .forms import CourseForm**

**def create\_course(request):**

**if request.method == 'POST':**

**form = CourseForm(request.POST)**

**if form.is\_valid():**

**form.save()**

**return redirect('course\_list') # Przekierowanie do listy kursów**

**else:**

**form = CourseForm()**

**return render(request, 'create\_course.html', {'form': form})**

tworzymy nowy szablon o nazwie create\_course.html w course/templates

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Nowy Kurs</title>**

**</head>**

**<body>**

**<h1>Dodaj Nowy Kurs</h1>**

**<form method="post">**

**{% csrf\_token %}**

**{{ form.as\_p }}**

**<button type="submit">Zapisz</button>**

**</form>**

**<a href="{% url 'course\_list' %}">Powrót do listy kursów</a>**

**</body>**

**</html>**

Do course/urls.py dodajemy

**path('new/', views.create\_course, name='create\_course'), # Widok tworzenia kursu**

Dodajemy też do course\_list.html taki fragment kodu:

<!DOCTYPE html>

<html>

<head>

<title>Lista Kursów</title>

</head>

<body>

**<a href=”{% url ‘create\_course’ %}”>Dodaj nowy kurs</a>**

<h1>Lista Kursów</h1>

<ul>

{% for course in courses %}

<li>

<h2>{{ course.title }}</h2>

<p>{{ course.content }}</p>

<p><strong>Autor:</strong> {{ course.author }}</p>

<p><em>Utworzono: {{ course.created\_at }}</em></p>

<p><em>Aktualizowano: {{ course.updated\_at }}</em></p>

</li>

{% endfor %}

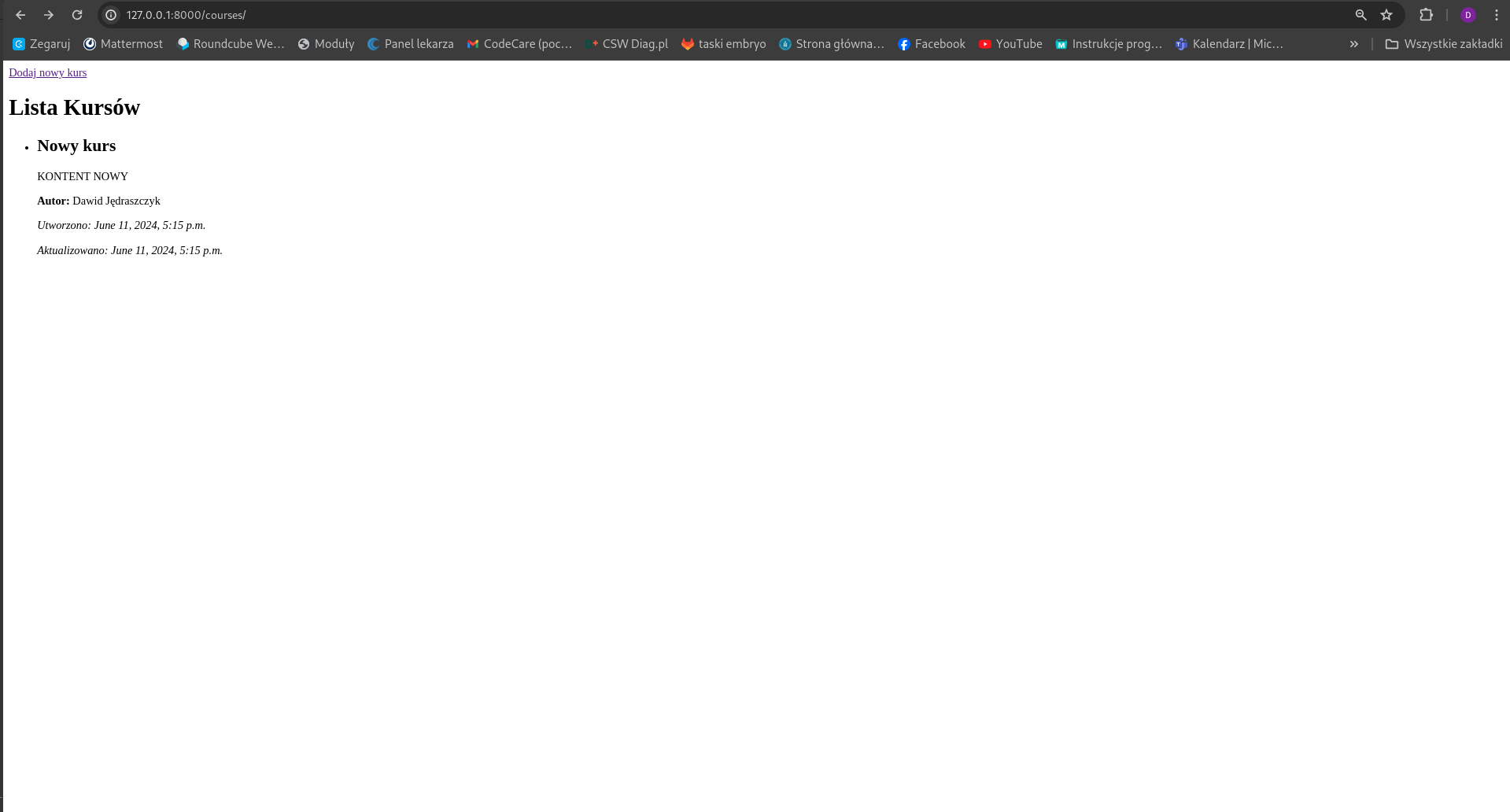
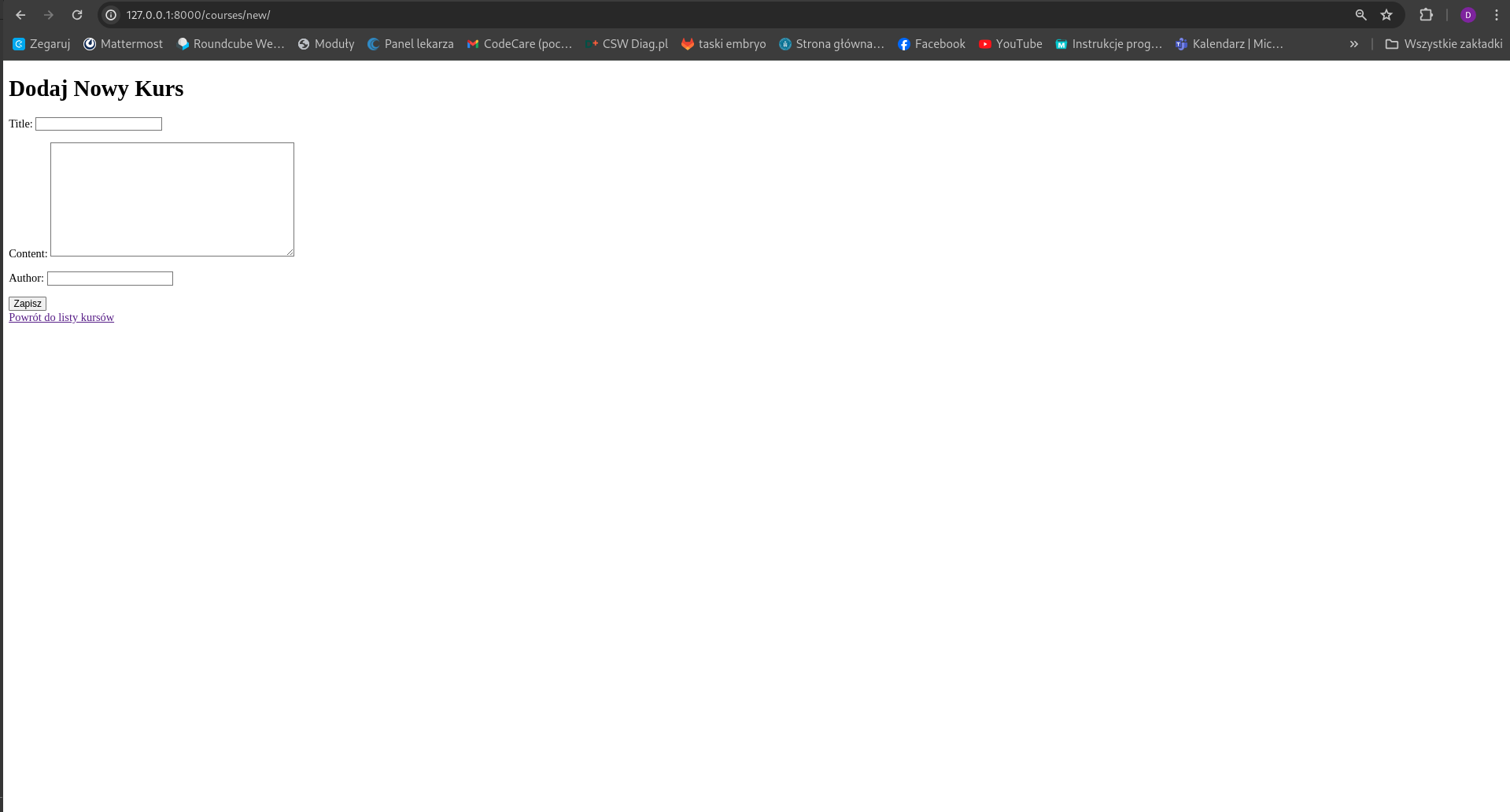
</ul>

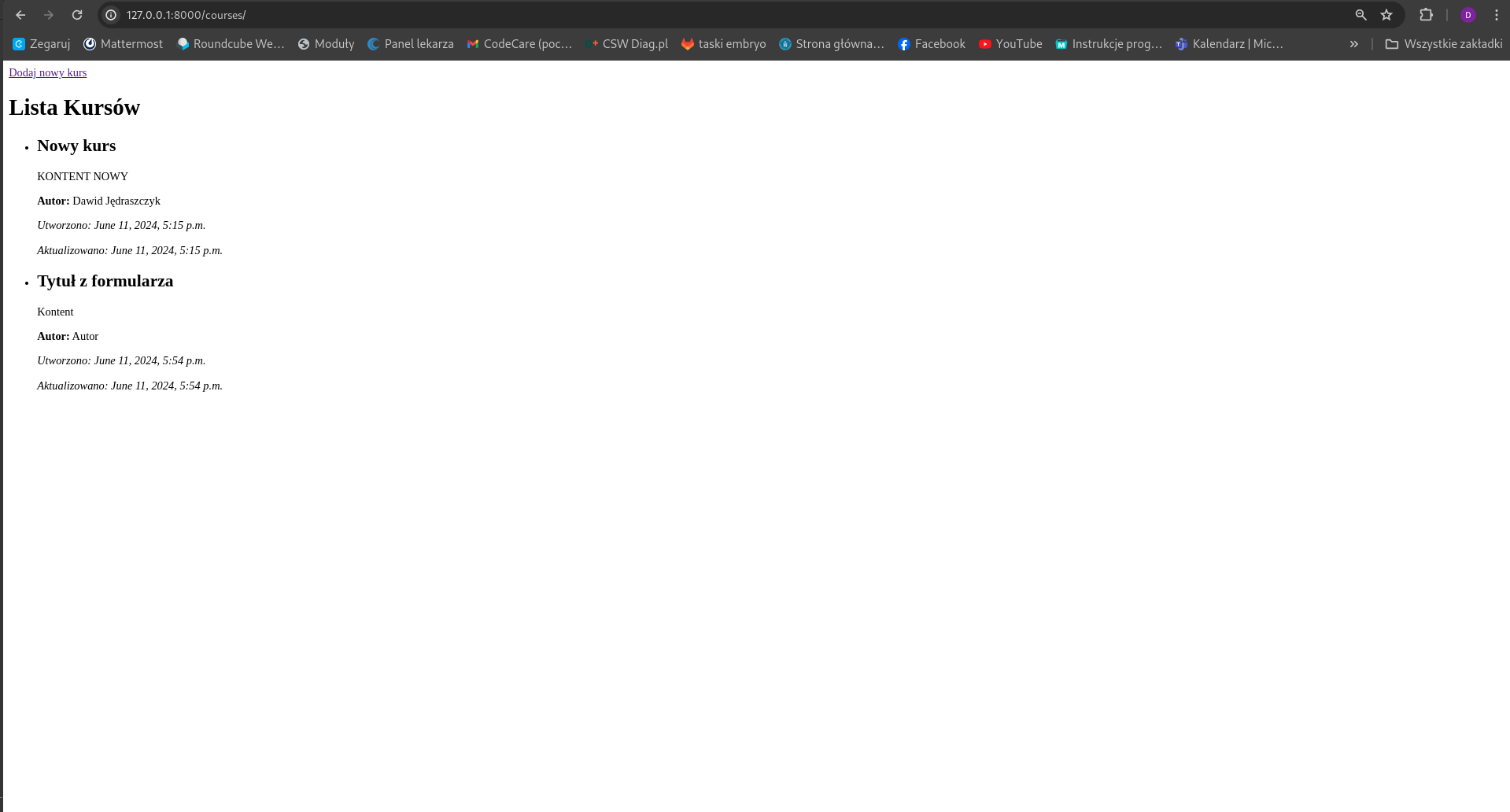
</body>

</html>

Odpalamy serwer!

Nowy widok listy kursów:

  
Widok formularza:  
  
Po wypełnieniu formularza jest tworzony nowy kurs:



8. Zadania do zrobienia samodzielnie

Jeżeli nie udało Ci się skonfigurować projektu, pobierz go z githuba:

**github.com/DawidJedraszczyk/django-introduction**

Po pobraniu, pozostanie Ci do zainstalowania biblioteka Django.

Pomocna będzie Ci dokumentacja django i stackoverflow.

[docs.djangoproject.com/en/5.0/](https://docs.djangoproject.com/en/5.0/)

Twoim zadaniem będzie:

* stworzenie widoku, który usuwa wybrany kurs
* stworzenie nowej aplikacji o nazwie Teacher

\*\* w aplikacji Teacher stwórz model Teacher o polach np. Imie nazwisko wiek,

\*\* w modelu Teacher dodaj course jako ForeignKey,

\*\* Obsłuż widoki listy, tworzenia i usuwania dla Teachera,