Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

кафедра Програмних засобів

**ЗВІТ**

з лабораторної роботи № 2

з дисципліни «Проектування сучасних інформаційних систем»

з теми: **«**Створення проекту Django та запуск сервера із Nginx**»**

Виконала:

студентка групи КНТ-219м А. В. Кущ

Прийняв:

к.т.н., доцент Г.В. Табунщик

2019

**1. Мета роботи**

Ознайомитися із основними можливостями фреймфорку Django, навчитися створювати та запускати проекти із використанням утиліти Supervisor та Nginx сервера.

**2. Завдання до роботи**

1. Застосувавши конспект лекцій та додаткову літературу, вивчити принципи створення проектів у Django.
2. Згідно з наведеним прикладом створити веб-застосування за індивідуальним завданням викладача.
3. Оформити звіт. Звіт з лабораторної роботи повинен містити: мету роботи; завдання на виконання роботи; тексти програм; результати роботи; висновки.
4. Відповісти на контрольні запитання.

**3. Результати роботи**

**1.** Для можливості роботи з різними проектами, що можуть використовувати різні бібліотеки та версії Python – створюємо віртуальне оточення за допомогою утиліти **virtualenv**. Після чого виконуємо активацію оточення, в результаті чого перед ім’ям користувача з’явиться назва віртуального середовища у дужках (рис.3.1).

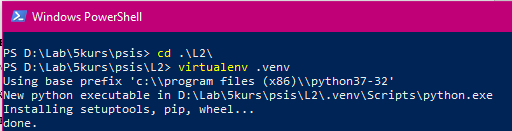




Рисунок 3.1 – Створенням та активація віртуального оточення

**2.** Виконуємо команди для створення Django-проекту (рис.3.2).

**pip install Django** – встановлюємо залежність Django

**django-admin.py startproject** **project** – створення проекту

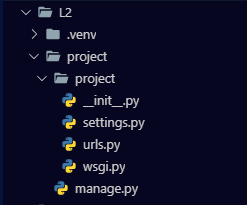


Рисунок 3.2 – Структура папок з проектом та самого проекту

**3**. Створення бази даних, для цього переходимо до каталогу із файлом manage.py та виконуємо команду.

**python manage.py migrate** – створення бази даних (рис.3.3).

За замовчуванням в Python використовується SQLite база даних. Команда **migrate** аналізує значення **INSTALLED\_APPS** і створює всі необхідні таблиці в базі даних, використовуючи налаштування бази даних з файлу **settings.py** та міграції з застосування.

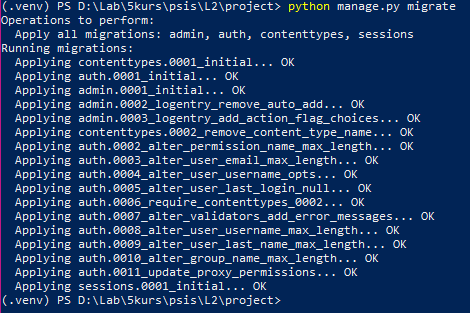


Рисунок 3.3 – Створення бази даних

**4.** Для того, щоб перевірити правильність первинних налаштувань, необхідно виконати команду **python manage.py runserver** та перейти за посиланням http://127.0.0.1:8000/ на стандартну сторінку Django (рис.3.4-3.5).

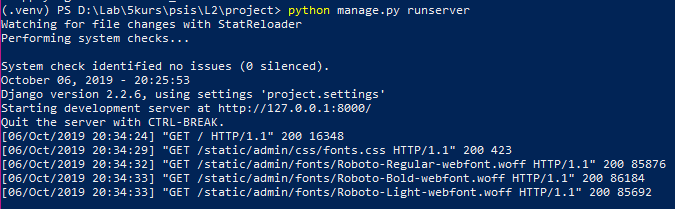


Рисунок 3.4 – Результат виконання команди

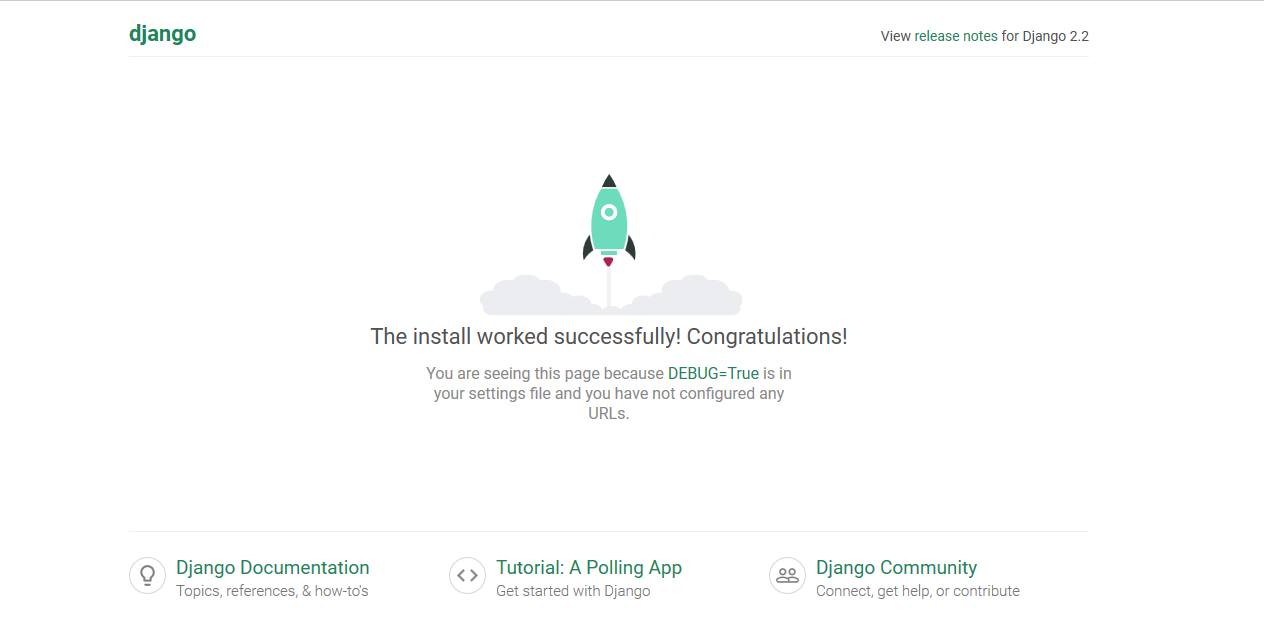


Рисунок 3.5 – Сторінка Django

**5.** Створення застосування з назвою **blog** виконується за допомогою команди

**python manage.py startapp blog.** Для зручності виконано перейменування каталогу project на main, у якому містяться налаштування Django за замовчуванням (рис3.6).

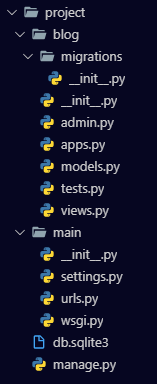


Рисунок 3.6 – Структура проекту після створення застосування blog

**6.** Створене застосування необхідно додати у файл налаштувань **main/settings.py** (рис3.7).

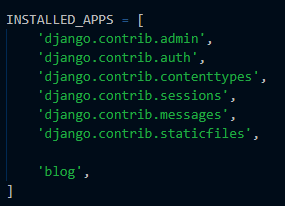


Рисунок 3.7 – Додавання застосунку blog

**7. Створення моделей та бази даних.**

**7.1.** Що створити модель необхідно відредагувати файл **blog/models.py** (рис3.8).

Створена модель представлена класом Article, успадкованим від **django.db.models.Model** та містить два атрибути **title** (назва) та **description** (опис), які є екземплярами класу **Field** та полями в таблиці бази даних. Методом **\_\_str\_\_** повертає строкове представлення об’єкту.

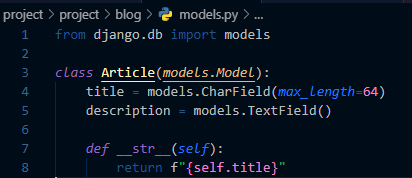


Рисунок 3.8 – Створення моделі

8. Виконання міграції. Оскільки таблиці бази даних в Django генеруються на основі моделей даних, що описуються класами Python, то після редагування файлу з моделями необхідно виконати **makemigrations** (рис.3.9).

**./manage.py** **makemigrations** – створення міграції.

Виконання команди дозволяє від слідити зміни та зберегти їх до міграцій, які у свою чергу є просто файлами, що зберігаються на диску та містять інформацію про внесені зміни та про структуру бази даних.

**python manage.py migrate** - команда **migrate** виконує міграції і автоматично оновлює базу даних.

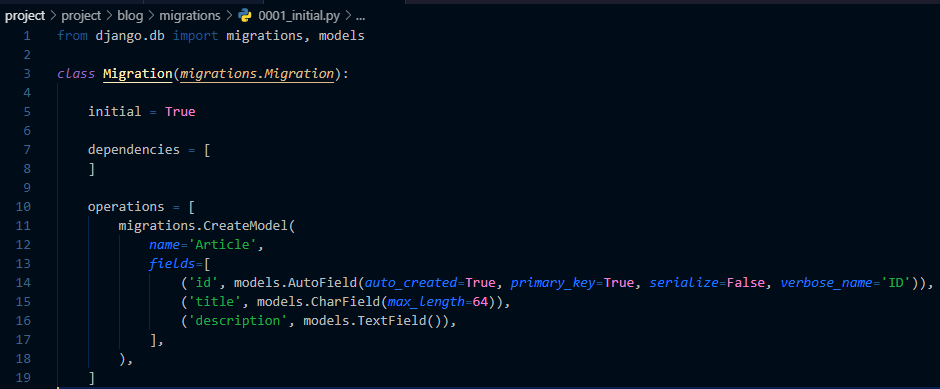


Рисунок 3.9 – Файл міграції

**9. HTML шаблони Django.**

**9.1.** Створення каталогу. Для використання шаблонів необхідно створити каталог **templates** в кореневому каталогу проекту поруч із manage.py та внести зміни до файлу налаштувань **(./main/settings.py)** в змінній **TEMPLATES** за ключем **DIRS** (рис.3.10).

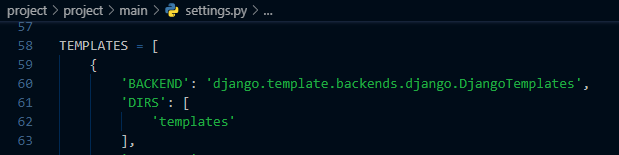


Рисунок 3.10 – Налаштування для каталогу шаблонів

**9.2.** Створення HTML – сторінок. Додання 3 шаблонів до каталогу **templates:**

**create.html,** **index.html, main.html.**

В шаблоні **main.html** використовується ітератор for, який перебирає масив з об’єктів Article Доступ до змінної здійснюється за допомогою **«{{ }}»,** також тут використовується метод \_\_**str**\_\_, для відображення об’єктів, тобто відображає створенні записи блогу (рис.3.11).



Рисунок 3.11 – Зміст шаблону main.html

**index.html** є шаблоном, що містить підключення Bootstrap та містить тег <style> для відображення зовнішнього вигляду сторінки (рис.3.12).



Рисунок 3.12 – Зміст шаблону index.html

Шаблон **create.html** містить тег <**style**> для відображення зовнішнього вигляду форми для створення заголовку та опису блогу (рис.3.13).

До шаблону додається елемент **form** із атрибутом **action**, що визначає на яку адресу буде відправлений запит, та **method** — яким методом.

У випадку створення нового об’єкту використовується запрос типу POST**.  
{% url <url\_name> %}** - це шаблонний тег, що дозволяє отримати повну адресу за ім’ям цієї адреси у структурі URL Django.

Django забезпечує захистом проти Cross Site Request Forgeries. Для відправки форми через POST необхідно використовувати шаблонний тег **csrf\_token**.

Поле **input** з типом **submit** є фактично кнопкою, за допомогою якої користувач підтверджує відправлення даних до серверу.



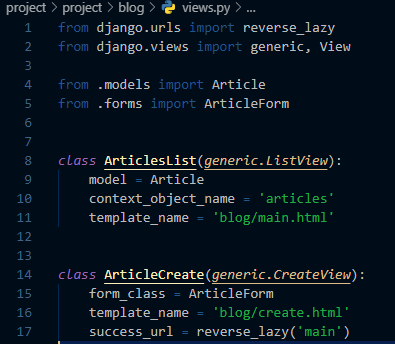
Рисунок 3.13 – Зміст шаблону create.html

**10.** **Створення View у Django.** Виконати редагування файлу **./blog/views.py** – це файл, який міститиме представлення, які використовуються для роботи з базою або для виконання певних задач (рис.3.11). В Django є безліч класів, що спрощують процес розробки проектів, у даному файлі їх два: **ArticlesList** та **ArticleCreate**.

Клас **ArticlesList** наслідує клас **ListView**, що призначений для виведення списків об’єктів. Він дозволяє отримати об’єкти, модель яких вказана у атрибуті **model** в даному випадку це **Article**, та використовувати у HTML шаблоні за допомогою змінної у атрибуті **context\_object\_name**. Атрибут **template\_name** дозволяє вказати назву шаблону, що буде використовуватися.

Клас **ArticleCreate** наслідує клас **CreateView**, та дозволяє показати користувачу створену форму (рис.3.13). **CreateView** — клас, призначений для обробки запитів на створення нових об’єктів бази даних. Для роботи з формою треба вказати **form\_class**, в даному випадку це **ArticleForm** (./**blog/forms.py**), та назву шаблона.

Функція **reverse\_lazy** дозволяє **Django** отримати **url** адресу за назвою, що вказана у змінній name URL шаблону, щоб уникнути написання повної адреси та помилок у разі внесення змін до адреси. **success\_url** визначає url, на яку буде перенаправлений користувач у разі успішного створення об’єкту.

  
Рисунок 3.11 – Зміст файлу views.py

**11. URL адреса.**

У файлі **blog**/**urls.py**, міститься список посилань, за якими викликаються необхідні посилання та їх методи (рис.3.12).

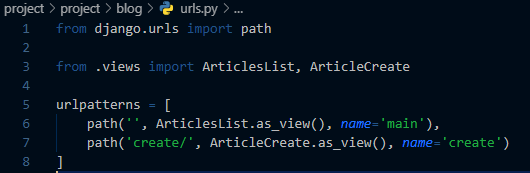


Рисунок 3.12 – Зміст файлу urls.py

**12. Форми**

Реалізований у в Django класу Form, дозволяє описати форму, як вона працює та відображається для користувача. Для цього до каталогу blog необхідно додати файл forms.py (рис. 3. 13). Файл містить клас ArticleForm та клас Meta, вказавши в   
model = Article, таким чином створивши форму для моделі Article, вибравши при цьому усі атрибути 'title', 'description'.

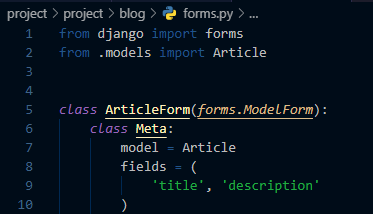


Рисунок 3.13 – Зміст файлу form.py

Для того, щоб форму можна було показати користувачу у файл **./blog/views.py** додано клас **ArticleCreate**, який буде працювати із формою.

Результат розробки блогу (рис.3.14).

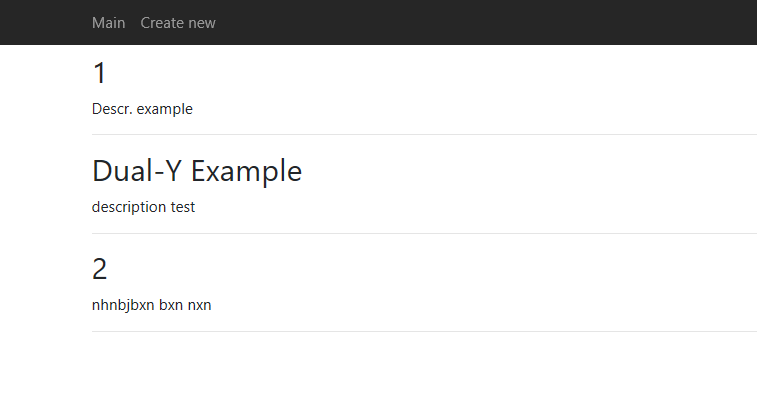


Рисунок 3.14 – Результат

Щоб мати можливість редагування/видалення записів можно скористатись admin-panel, що вбудована в Django, для цього необхідно створити супер-користувача за допомогою команди **python3 manage.py createsuperuser** та запустити сервер  
(рис. 3.15).

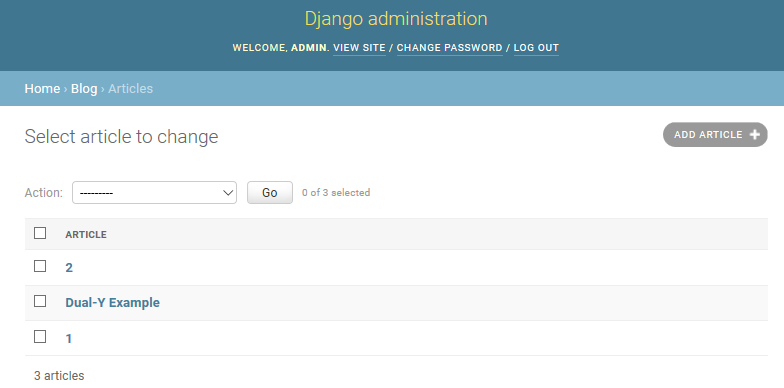
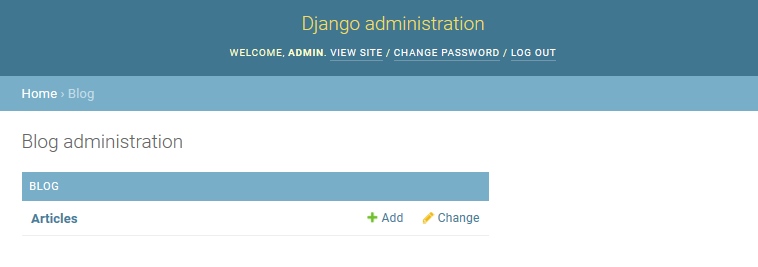


Рисунок 3.15 – Адмін-панель

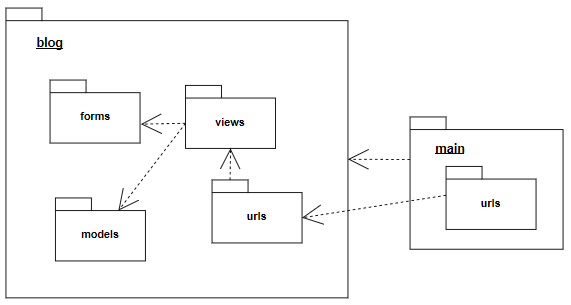


Рисунок 3.16 –Діаграма пакетів

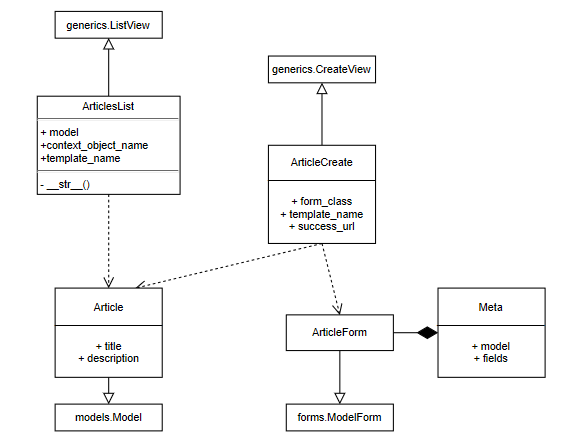


Рисунок 3.17 –Діаграма класів

**4. Висновок**

На цій лабораторній роботі я ознайомилась із основними можливостями фреймфорку Django, навчилася створювати та запускати проекти із використанням утиліти Supervisor та Nginx сервера.