Київський національний університет імені Тараса Шевченка

ННІ «Інститут геології»

Кафедра геоінформатики

КУРСОВА РОБОТА

на тему: «Створення та відображення шейп-файлів з використанням ArcGis API, бібліотеки ArcPy та фреймворку Django з підключеною базою даних SQLite»

Виконав студент 3-го курсу

групи геоінформатики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Земляков М. Р.

(підпис)

Науковий керівник:

кандидат ф.-м. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демидов В.К

(підпис)

Завідувач кафедри:

Доктор технічних наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Зацерковний В. І.

(підпис)

Київ 2017

# Зміст

[Зміст 2](#_Toc483406567)

[Вступ 3](#_Toc483406568)

[1.Теоретичні відомості 4](#_Toc483406569)

[1.1 Шейп-файли 4](#_Toc483406570)

[1.1.1 Формат шейп-файлу (.shp) 6](#_Toc483406571)

[1.1.2 Формат індексного файлу (.shx) 7](#_Toc483406572)

[1.1.3 Формат атрибутивного файлу (.dbf) 7](#_Toc483406573)

[1.2 Фреймворк Django 8](#_Toc483406574)

[1.2.1 Поняття Фреймворк 8](#_Toc483406575)

[1.2.2 Django 8](#_Toc483406576)

[1.3 Python і модуль ArcPy 13](#_Toc483406577)

[1.3.1 Загальні відомості про Python 13](#_Toc483406578)

[1.3.2 Модуль ArcPy 15](#_Toc483406579)

[1.4 База даних SQLite 19](#_Toc483406580)

[1.5 GitHub 21](#_Toc483406581)

[2. Практична частина 23](#_Toc483406582)

[2.1 Створення форми завантаження шейпу. Спосіб 1 23](#_Toc483406583)

[2.2 Створення форми завантаження шейпу. Спосіб 2 24](#_Toc483406584)

[Висновок 27](#_Toc483406585)

[Список використаної літератури 28](#_Toc483406586)

[Додатки 29](#_Toc483406587)

# Вступ

Сьогодні, як і вчора, як і завтра залишається актуальною наука про ГІС, бо мало яка область діяльності людини не може бути оптимізована, покращена за допомогою поверхневого чи глибокого аналізу геоданих. Часто на перший погляд важко інтерпретувати дані, зрозуміти їх суть та вилучити з них користь. Якщо мова йде про геодані, то кращим інструментом їх обробки та візуалізації є геоінформаційні програмні продукти, в нашому випадку ArcGis. За допомогою цієї програми можна дослідити дані візуально, провести їх класифікацію чи змінити їх. Проблема цього підходу полягає у двох речах. Першою проблемою є те, що програмний продукт не безкоштовний і коштує не дешево. Друга проблема – це відсутність чи недостатня кваліфікація потенціального користувача , що унеможливлює користування програмним забезпеченням. І якщо перша проблема вирішується зміною ПЗ на те, яке вільно розповсюджується(QGIS), то з другою все складніше.

Метою моєї роботи є систематизація та закріплення теоретичних знань, що були отримані під час проходження курсу програмування у ГІС.

Завданням роботи є створення функціоналу на сайті, яке дозволить без спеціальної кваліфікації створювати шейп-файли за запитом та додавати їх на карту для подальшого візуального аналізу.

# 1.Теоретичні відомості

## 1.1 Шейп-файли

«Шейп-файл» (англ. Shapefile) — популярний векторний формат файлів для програмного забезпечення геоінформаційної системи. Розробляється та підтримується компанією ESRI з метою сумісності даних між продуктами ESRI та стороннім програмним забезпеченням. Формат файлу дозволяє зберігати наступні типи геометричних фігур: точки, лінії, які представляють, для прикладу дороги та стежки, полігони, що представляють, наприклад, колодязі, річки, озера. Кожен елемент, як правило, має атрибути, які описують його, такі як ім'я, температура чи тип дороги і її завантаженість.

Формат shapefile є цифровим векторним форматом для зберігання геометричних об'єктів та пов'язаної з ними атрибутивної інформації. Цей формат не має можливості зберігати топологічну інформацію. Формат шейп був введений з ArcView GIS версії 2 на початку 1990-х років. Тепер можна читати і писати географічні набори даних, використовуючи формат шейп файлу з широким спектром програмного забезпечення.

Формат шейп простий тому, що він може зберігати примітивні геометричні типи даних точок, ліній і полігонів.

Термін «шейп-файл» досить часто помилково асоціюють лиш з одним файлом. Насправді специфікація формату визначає набір файлів із загальним префіксом імені файлу, які зберігаються в тому ж каталозі . Обов'язковий файл .shp. Хоча деяке програмне забезпечення вимагає три файли: .shp, .shx і .dbf. Фактично шейп відноситься конкретно до файлу .shp. Застаріле програмне забезпечення ГІС може потребувати, щоб префікс імені файлу був обмежений вісьмома символами, щоб відповідати вимогам DOS для імені файлу Конвенції 8.3, хоча сучасні програми оперують файлами із довшими іменами.

Обов'язкові файли:

* .shp — головний файл; містить набори геометричних об'єктів
* .shx — індексний файл; використовується для зв'язку між файлами .dbf і .shp
* .dbf — атрибутивний файл; містить атрибути об'єктів, описаних в .shp файлі у форматі dBase IV

Інші файли

* .prj — формат проекції; системи і відомості про проекції, текстовий файл, що описує проекцію, використовуючи поширений текстовий формат координат
* .sbn і .sbx — просторовий індекс з особливостей
* .fbn і .fbx — просторовий індекс з особливостей, які доступні тільки для читання
* .ain і .aih — індекс атрибуту активних полів в таблиці
* .ixs — індекс геокодування для читання і запису наборів даних
* .mxs — індекс геокодування для читання і запису наборів даних (формат ОДБ)
* .atx — індекс атрибут файлу .dbf у вигляді шейп ColumnName .atx (ArcGIS 8 і пізніше)
* .shp.xml — геопросторові метадані у форматі XML, такому як ISO 19115 або інша XML-схема
* .cpg — використовується для вказівки кодової сторінки (тільки для .dbf) для ідентифікації кодувань, які будуть використовуватися
* .qix — альтернативний дерево-квадрантному просторовий індекс, що використовується в MapServer та програмному забезпеченні GDAL / OGR
* У кожному з .shp, .shx, і .dbf файлів, геометричні фігури в кожному файлі відповідають один одному в послідовності (тобто, перший запис у файлі відповідає .shp першому запису в .SHX і .dbf файлах і т. д.). .shp і .shx файли мають різні поля з різним порядком байтів, тому ті, хто випускає файли, повинні бути дуже обережні, щоб дотримуватися порядку байтів для кожного поля і розглядати його належним чином.

Формат шейп не має можливості зберігати топологічну інформацію. У ESRI ArcInfo використовуються різні БД, які дають можливість зберігати особливість топології.

### 1.1.1 Формат шейп-файлу (.shp)

Основний файл (.shp) містить геометричні дані. Двійковий файл складається з одного заголовка фіксованої довжини та одного або більше записів змінної довжини. Кожен із записів включає в себе заголовок запису та контент (як правило, набір координат точок). Детальний опис формату файлу дається в технічному описі шейп-файлу ESRI . [1] Цей формат не слід плутати з форматом джерела шрифту фігури AutoCAD, який також має розширення .shp .

Порядок 2D осей передбачає декартову систему координат, використовуючи порядок (XY) або (широта-довгота). Цей порядок осей відповідає географічній системі координат, де порядок аналогічний. Геометрія також може підтримувати 3 або 4 мірні Z і М координати, для підвищень та замірів, відповідно. Вимір Z зберігає висоту кожної координати в 3D просторі, які можуть бути використані для аналізу або для візуалізації геометрії з використанням 3D комп'ютерної графіки. Визначений користувачем вимір М може бути використаний для однієї з багатьох функцій, таких як зберігання замірів, чи часу в 4D просторі.

Основний заголовок файлу фіксований, довжина його становить 100 байт і містить він 17 полів; дев'ять 4-байтних (32-розрядне ціле число або int32) та вісім полів з плаваючою комою та із знаком.

### 1.1.2 Формат індексного файлу (.shx)

Індексний файл містить той же заголовок 100 байт як .shp файл, а потім будь-яке число 8 байтових записів фіксованої довжини, які складаються з двох полів.

Використовуючи цей індекс, можна шукати в зворотному напрямку по шейп-фйлу завдяки, по-перше, переміщенні назад в індексах фігур (що можливо, оскільки він використовує записи фіксованої довжини), далі читаючи зміщення запису, і за допомогою цього зміщення отримати правильне положення у .shp файлі.

### 1.1.3 Формат атрибутивного файлу (.dbf)

Атрибути для кожної фігури зберігаються у форматі DBase. Альтернативний формат, який також може бути використаний — XBase, який має відкриту специфікацію, і використовується в відкритих вихідних Shapefile бібліотек, таких як бібліотеки Shapefile C

## 1.2 Фреймворк Django

### 1.2.1 Поняття Фреймворк

Фреймворк (англ. Framework, каркас, платформа, структура, інфраструктура)—інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою.

Програмний фреймворк (англ. software framework) — це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми. Відповідно — програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

Одна з головних переваг, при використанні каркасних застосунків, полягає в тому, що такі програми мають стандартну структуру. Каркаси застосунків стали популярними з появою елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для додатків. З їх використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь. Для забезпечення каркасу, зазвичай, використовують підходи об'єктно-орієнтованого програмування, наприклад, частини програми можуть успадковуватися від базових класів фреймворка.

### 1.2.2 Django

Django (Джанго) — високорівневий відкритий Python-фреймворк для розробки веб-систем. Названо його було на честь джазмена Джанго Рейнхардта (відповідно до музичних смаків одного зі засновників проекту)

Сайт на Django будується з однієї або декількох частин, які рекомендується робити модульними. Це одна з істотних архітектурних відмінностей цього фреймворку від деяких інших (наприклад Ruby on Rails).

Архітектура Django подібна на «Модель-Вид-Контролер» (MVC). Однак, те що називається «контролером» в класичній моделі MVC, в Django називається «вид» (англ. view), а те, що мало б бути «видом», називається «шаблон» (англ. template). Таким чином, MVC розробники Django називають MTV («Модель-Шаблон-Вид»).

Початкова розробка Django, як засобу для роботи новинних ресурсів, досить сильно позначилася на його архітектурі: він надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Так, наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, в Django є вбудований модуль для керування вмістом, який можна включити в будь-який сайт, зроблений на Django, і який може керувати відразу декількома сайтами на одному сервері. Адміністративний модуль дозволяє створювати, змінювати і вилучати будь-які об'єкти наповнення сайту, протоколюючі всі дії, а також надає інтерфейс для управління користувачами і групами (з призначенням прав).

У дистрибутив Django також включені програми для системи коментарів, синдикації RSS і Atom, «статичних сторінок»(якими можна управляти без необхідності писати контролери та відображення), перенаправлення URL та інше.

Django був створений для управління сайтами новин LJWorld.com, lawrence.com і KUsports.com компанії The World Company (Лоренс (Канзас), США), але з моменту початку розповсюдження його у статусі відкритого програмного забезпечення отримав величезну популярність в усьому світі як платформа до численних систем.

#### 1.2.2.1 Деякі можливості Django

* ORM, API доступу до БД з підтримкою транзакцій

Django підтримує парадигму ООП. Об'єкти БД в термінології Django іменуються «моделями». Фреймворк надає у розпорядження розробникові розвинутий прикладний програмний інтерфейс для високорівневого доступу до даних. В більшості випадків немає потреби писати SQL-запити (однак, ніхто не забороняє це робити).

Для прикладу, для проекту обліку учнів можна створити таку модель:

class Student(models.Model):

name = models.CharField("Ім'я", max\_length="100")

surname = models.CharField("Прізвище", max\_length="100")

birth\_date = models.DateField()

* вбудований інтерфейс адміністратора, з уже наявними перекладами на більшість мов

Django автоматично створить для вас CRUD-інтерфейс ('адмінку').

* диспетчер URL на основі регулярних виразів

Парсинг URL-ів побудований на регулярних виразах. Розробник не обмежений у використанні певної схеми посилань

* розширювана система шаблонів з тегами та наслідуванням

В Django є окрема мова для опису шаблонів. Вона є дуже простою і «дружньою» для непрограмістів. В ній присутні оператори циклу, умови, форматування даних. З практики: верстальщика можна навчити редагувати Django-шаблони за 2-3 заняття.

Мова шаблонів виконує функцію відображення даних. Змінити дані в БД її операторами неможливо.

* система кешування

Django-проект може бути налаштований на роботу з Memcached чи будь-яким іншим фреймворком за лічені хвилини. Інструменти Django дозволяють кешувати SQL-вибірки, шаблони та їх частини і просто окремі змінні.

* Інтернаціоналізація

Переклад Django-проекту не є проблемою. Інтернаціоналізація базується на концепції «лінивого» перекладу. Це зокрема означає, що якщо певний рядок тексту не має перекладу, то буде використано базовий текст і не буде показано повідомлення про помилку. Проте ніхто не забороняє використовувати функції, які будуть контролювати наявність перекладу рядкових даних.

* архітектура застосунків, що підключаються, які можна встановлювати на будь-які Django-сайти
* «generic views» - шаблони функцій контролерів
* авторизація та аутентифікація, підключення зовнішніх модулів аутентифікації: LDAP, OpenID та ін.
* система фільтрів («middleware») для побудови додаткових обробників запитів, наприклад включені в дистрибутив фільтри для кешування, стиснення, нормалізації URL і підтримки анонімних сесій
* бібліотека для роботи з формами (наслідування, побудова форм за існуючою моделлю БД)
* вбудована автоматична документація по тегам шаблонів та моделям даних, доступна через адміністративний за стосунок

## 1.3 Python і модуль ArcPy

### 1.3.1 Загальні відомості про Python

Python (найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Серед основних її переваг можна назвати такі:

* чистий синтаксис (для виділення блоків слід використовувати відступи);
* переносність програм (що властиве більшості інтерпретованих мов);
* стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
* можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
* стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
* зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
* відкритий код (можливість редагувати його іншими користувачами).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python також зручна як мова розширення для прикладних програм, що потребують подальшого налагодження.

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з рядками, зокрема, кодованими в юнікоді.

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви) і від версії 2.4, множини.

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, й за необхідності можливе успадкування навіть від базових типів.

### 1.3.2 Модуль ArcPy

ArcPy - це пакет сайту Python, який забезпечує успішний і продуктивний аналіз географічних даних, конвертацію даних, управління даними і автоматизацію карт в Python.

ArcPy забезпечує доступ до інструментів геообробки, а також до додаткових функцій, класів і модулів, які дозволяють швидко створювати як прості, так і складні робочі процеси.

Додатки та скрипти ArcGIS написані з використанням ArcPy, що дозволяє отримати доступ до численних модулів Python, які розробили користувачі ГІС і програмісти, що працюють в різних галузях. Ще одна перевага використання ArcPy в середовищі Python полягає в тому, що Python є універсальною мовою програмування, який легко освоїти. Це мова, що інтерпретується з динамічною типізацією, що дає можливість швидко моделювати і перевіряти скрипти в інтерактивному середовищі і разом з тим забезпечує можливість написання великих додатків.

З технічної точки зору інструменти геообробки представляють собою функції з arcpy, тобто, доступ до них здійснюється так само, як і до будь-яких інших функцій Python. Проте, щоб уникнути плутанини між інструментами і відмінними від них функціями (такими як службові функції на зразок ListFeatureClasses ()), існують чіткі відмінності.

* Інструменти документуються не так, як функції. У кожного інструменту є власна довідкова сторінка в довідковій системі. Функції документуються в документації ArcPy.
* Інструменти, на відміну від функцій, повертають об'єкт Result.
* Інструменти створюють повідомлення, до яких можна звертатися за допомогою багатьох функцій, таких як GetMessages (). Функції не створюють повідомлень.

#### 1.3.2.1 Курсор у ArcPy

Курсор - це об'єкт доступу до даних, які можуть бути використані як для ітерації по набору рядків в таблиці, так і для вставки нових рядків у таблицю. Курсори можуть бути трьох форм: пошуку, вставки або оновлення. Як правило, курсори використовуються для зчитування і поновлення атрибутів.

Приклад курсору для пошуку:

import arcpy

# Set the workspace

arcpy.env.workspace = "c:/base/data.gdb"

# Create the search cursor

cursor = arcpy.SearchCursor("roads", '"TYPE" <> 4')

# Iterate through the rows in the cursor

for row in cursor:

print("Name: {0}, CFCC code: {1}".format(row.NAME, row.CFCC))

del cursor, row

Приклад курсору для оновлення:

import arcpy

# Set the workspace

arcpy.env.workspace = "c:/base/data.gdb"

# Create the update cursor

cursor = arcpy.UpdateCursor("roads")

# Update the road buffer distance field based on road type.

# Road type is either 1,2,3,4 Distance is in meters.

for row in cursor:

row.setValue("BUFFER\_DIST", row.getValue("TYPE") \* 100)

cursor.updateRow(row)

# Delete cursor and row objects

del cursor, row

Приклад курсору для вставки:

import datetime

import arcpy

# Create insert cursor for table

cursor = arcpy.InsertCursor("c:/base/data.gdb/roads\_maint")

# Create 25 new rows. Set default values on distance and CFCC code

for i in range(1000, 1025):

row = cursor.newRow()

row.setValue('rowid', i)

row.setValue('distance', 100)

row.setValue('CFCC', 'A10')

row.setValue('LastInsp', datetime.datetime.now())

cursor.insertRow(row)

# Delete cursor and row objects

del cursor, row

## 1.4 База даних SQLite

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92.. Сирцевий код SQLite поширюється як суспільне надбання (англ. public domain), тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Фінансову підтримку розробників SQLite здійснює спеціально створений консорціум, до якого входять такі компанії, як Adobe, Oracle, Mozilla, Nokia, Bentley і Bloomberg.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.

Завдяки архітектурі рушія можливо використовувати SQlite як на вбудовуваних (embedded) системах, так і на виділених машинах з гігабайтними масивами даних.

Приклад коду для роботи з БД:

CREATE TABLE comments (

post\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

email TEXT NOT NULL,

website\_url TEXT NULL,

comment TEXT NOT NULL );

## 1.5 GitHub

GitHub — один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Сервіс безкоштовний для проектів з відкритим вихідним кодом, з наданням користувачам усіх своїх можливостей (включаючи SSL), а для окремих індивідуальних проектів пропонуються різні платні тарифні плани.

Розробники сайту називають GitHub «соціальною мережею для розробників».

Окрім розміщення коду, учасники можуть спілкуватись, коментувати редагування один одного, а також слідкувати за новинами знайомих. За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії — GitHub дає зручний інтерфейс для цього і може показувати вклад кожного участника в вигляді дерева.

Для проектів є особисті сторінки, невеликі Вікі та система відстеження помилок. Прямо на сайті можна дивитись файли проектів з підсвічуванням синтаксису для більшості мов програмування.

* На платних тарифних планах можна створювати приватні репозиторії, доступні обмеженому колу користувачів.
* Є можливість прямого додавання нових файлів в свій репозиторій через веб-интерфейс сервісу
* Код проектів можна не лише скопіювати через Git, але й завантажити у вигляді архіва. (Для цього достатньо додати /zipball/master/ в кінець адресного рядка.)
* Окрім Git, сервіс підтримує отримання і редагування коду через SVNи Mercurial.
* На сайті є pastebin-сервіс gist.github.com для швидкої публікації фрагментів коду.

Раніше Ruby-проекти могли бути автоматично опубліковані в RubyGems-репозиторії сервісу, але в жовтні 2009 GitHub відмовився від цього сервісу.

# 2. Практична частина

Практична частина полягає в розширенні функціоналу раніше створеного сайту з використанням наявних дата сетів(шейп-файли; дані, які на початок роботи ще не завантажені до БД) за допомогою можливостей фреймворку Django, бібліотеки ArcPy та ArcGis API.

Процесс створення сайту в даній практичній роботі опущено.

## 2.1 Створення форми завантаження шейпу. Спосіб 1

У файлі view.py, який підключено до проекту, створюємо функцію, яка буде обробляти запит, що прийшов від користувача.

На вхід подається request, з якого методом GET ми отримали дані з форми. Форма прописана в html файлі.

Задаємо воркспейс, дозволяємо перезапис файлів, створюємо просторовий об’єкт, де завдяки GET вже є параметр вибірки даних, зберігаємо його, архівуємо та подаємо користувачу на завантаження.

У файлі urls.py задаємо url, який при виклику через пошуковий рядок браузера користувача активує функцію у view.py.

І, звісно, не можна забувати про html файл, де прописано форму, яка отримує введені користувачем данні, а також викликає роботу url’а.

## 2.2 Створення форми завантаження шейпу. Спосіб 2

Другий спосіб створення і завантаження шейп-файлу відрізняється від першого тим, що джерелом даних є не інший шейп-файл, а база SQLite, що підключена до проекту. Плюсом цього способу є те, що можна додавати дані через панель адміна, що реалізована у Django, але для цього треба створити модель в файлі models.py.

Отже, створюємо клас у файлі models.py, що наслідує models.Model. Після команд makemigrations i migrate створиться таблиця в БД з вказаними у класі характеристиками.

В файлі admin.py регіструємо модель для доступу до неї через адмінку.

З файлом view.py проводимо ті ж дії, що й при першому способі, але тепер данні за допомогою запису ‘cur.execute("SELECT X,Y,type FROM gis\_points WHERE type = '%s'" % valuez)’ ми отримуємо з бази данних.

Також у цьому способі для заповнення таблиці шейп-файлу використовується arcpy.da.InsertCursor(). Інші строки коду залишаються без принципіальних змін.

Для повноцінної роботи моєї форми треба заповнити таблицю в базі даних, де Х та Y це координати точки, а type – її характеристика. Координати та характеристики точок ми отримали з існуючого шейп-файлу за допомогою arcpy.da.SearchCursor() і внесли у базу даних, виконавши цю строку «cur.execute('''INSERT INTO gis\_points (X, Y, type)

VALUES ( ?, ?, ? )''', ( x, y, z, ) )»

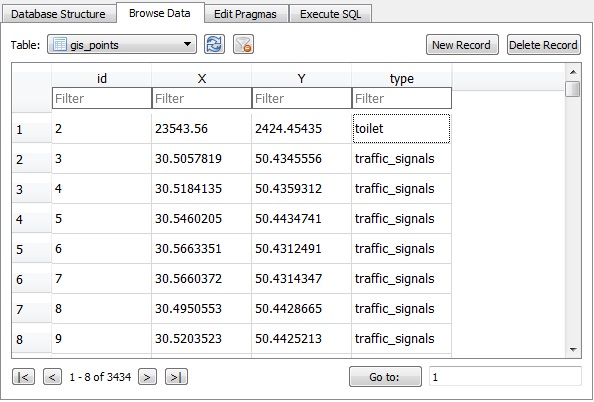


Рис.2.2.1 Структура БД

Така структура дозволяє легко отримати координати точки та помістити її у стовпець 'SHAPE@XY', який за замовчуванням присутній у шейп-файлі.

Коли весь код написаний і працює, а база даних заповнена, робимо add 🡪 commit 🡪 pull на GitHub.

Посилання на проект: https://github.com/Pickausaname/DjangoGIS

Найпростіший варіант працюючих форм виглядає таким чином:

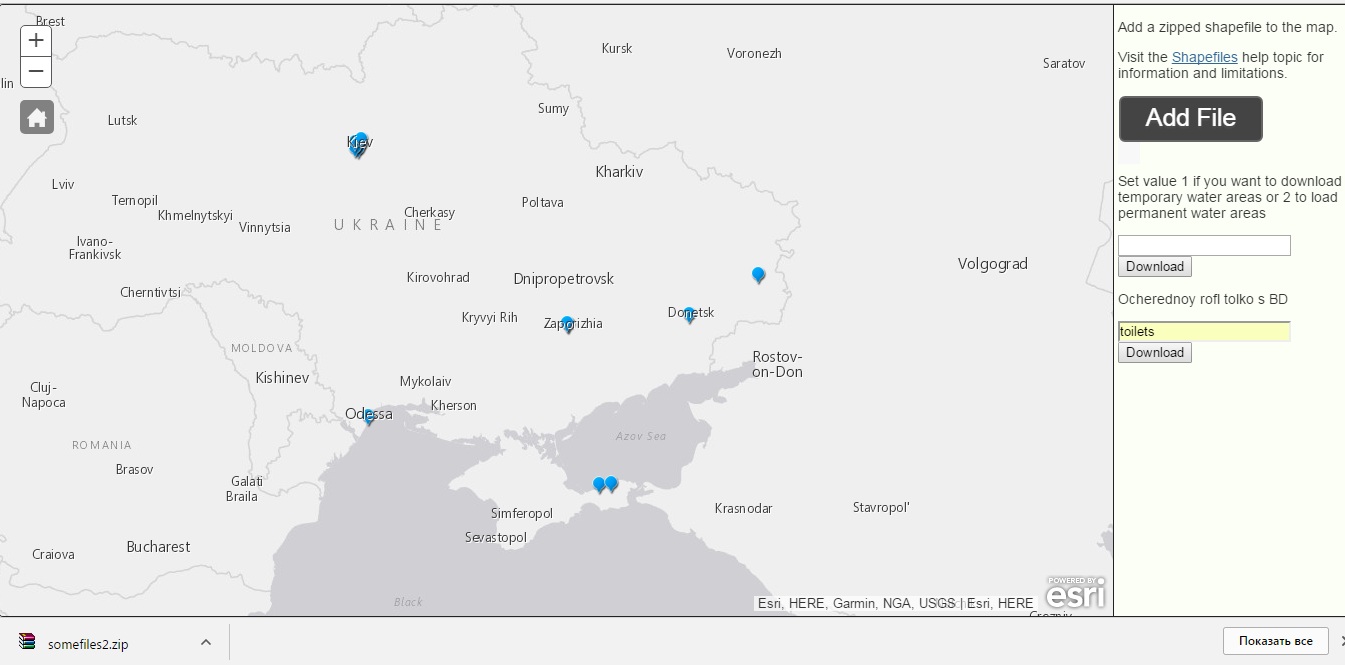


Рис.2.2.2 Зображення функціоналу

На рисунку можна побачити, що заповнення форми призводить до завантаження бажаного шейп-файлу у Zip архіві, який на цій же сторінці можна додати до карти. Можливість додавання архіву до карти реалізована розробниками ArcGis API.

Тепер, коли точки додані на карту, можна зайнятися візуальним аналізом карти.

# Висновок

Протягом написання цієї роботи сформувалося уявлення про потенційні можливості обробки геоданих за допомогою ArcPy у зв’язці з веб API.

Дякуючи Python on Django було створено веб-портал, який став базою для розміщення власних напрацювань. Під час роботи з ним я зміг виділити багато плюсів.

* Простота створення базового порталу
* Зв’язка з Python
* Логічна і не перевантажена структура проекту

Бібліотека ArcPy подарувала незабутні враження при роботі з нею.

Розробник забезпечив доступ до російськомовної документації, що дуже полегшило вивчення функціоналу, а навчальні приклади взагалі безцінні.

Єдине, що викликало труднощі - ArcGis API.

ArcGis API теж, безперечно, гарний програмний продукт. Розробники забезпечили опис функціоналу та таку приємну річ, як тест коду у sandbox, а робота з ним полягає у ctrl+c та ctrl+v.

Неприємною частиною роботи стало те, що просто шейп-файл додати на карту не можна, тому і було вирішено створити для користувача можливість завантажити файл спочатку на комп’ютер, а далі через API відобразити на екрані.

Під кінець цієї роботи хотілося б відзначити те, що її написання дуже добре допомогло мені систематизувати теоретичні знання з програмування, а також уявити продукти якого рівня можна створювати за допомогою використаних програм, бібліотек, додатків та фреймворків.

# Список використаної літератури

1. Django: стаття [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Django

2. Shapefile: стаття [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Shapefile

3. ArcGis: офіційний веб-портал [Електронний ресурс] – Режим доступу до ресурсу: http://www.esri.com/arcgis/about-arcgis

4. SQLite: Портал про БД [Електронний ресурс] – Режим доступу до ресурсу: http://www.freesource.info/wiki/SQLite

5. Cursor: стаття [Електронний ресурс] – Режим доступу до ресурсу: https://pro.arcgis.com/ru/pro-app/arcpy/classes/cursor.htm

6. Functional Programming HOWTO: документація [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/howto/functional.html

7. Python: стаття [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python

8. SQLite: стаття [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/SQLite>

# 

# Додатки

Додаток 1

Код файлу views.py

from django.http import HttpResponse

from django.shortcuts import render, get\_object\_or\_404, render\_to\_response

import arcpy, zipfile, StringIO

from gis.models import Articles

import os

import sqlite3

def home(request):

# return HttpResponse('Hello, world')

articles = Articles.objects.all()

context = {

'articles': articles

}

return render(request, 'gis/home.html', context)

def about(request):

return render(request, 'gis/about.html')

def show\_articles(request, article\_id):

article = get\_object\_or\_404(Articles, id=article\_id)

return render(request, 'gis/article.html', {'article': article})

def mymap(request):

return render(request, 'gis/test.html')

def search\_form(request):

return render\_to\_response('gis/search\_form.html')

def search(request):

#if 'q' in request.GET:

arcpy.env.workspace = r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\myfgdb.gdb"

arcpy.env.overwriteOutput = True

value=str(request.GET['q'])

arcpy.MakeFeatureLayer\_management(r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\myfgdb.gdb\hydro", "Copyhydro", '"TYPE\_LUT"='+ value )

arcpy.CopyFeatures\_management('Copyhydro', r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.shp')

arcpy.Delete\_management("Copyhydro")

filenames = [r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.dbf", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.prj", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.sbn", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.sbx", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.shp", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.shp.xml", r"C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\rofl\rofle.shx"]

zip\_subdir = "somefiles"

zip\_filename = "%s.zip" % zip\_subdir

s = StringIO.StringIO()

zf = zipfile.ZipFile(s, "w")

for fpath in filenames:

# Calculate path for file in zip

fdir, fname = os.path.split(fpath)

zip\_path = os.path.join(zip\_subdir, fname)

# Add file, at correct path

zf.write(fpath, zip\_path)

zf.close()

# Grab ZIP file from in-memory, make response with correct MIME-type

resp = HttpResponse(s.getvalue(), content\_type="application/x-zip-compressed")

# ..and correct content-disposition

resp['Content-Disposition'] = 'attachment; filename=%s' % zip\_filename

return resp

def search2(request):

valuez = str(request.GET['w'])

con = sqlite3.connect(r'C:\Users\S1mple\Desktop\djangoGIS\db.sqlite3')

cur = con.cursor()

arcpy.env.workspace = r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\shape2'

arcpy.env.overwriteOutput = True

arcpy.MakeFeatureLayer\_management(r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\shape2\kek.shp', 'Copykek')

arcpy.CopyFeatures\_management('Copykek', r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.shp')

arcpy.Delete\_management("Copykek")

cur.execute("SELECT X,Y,type FROM gis\_points WHERE type = '%s'" % valuez)

cursor = arcpy.da.InsertCursor(r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.shp', ['SHAPE@XY', 'type'])

for row in cur:

a = ((row[0], row[1]), '%s' % row[2])

cursor.insertRow(a)

con.close()

del cursor

filenames2=[r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.dbf',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.prj',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.sbn',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.sbx',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.shp',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.shp.xml',r'C:\Users\S1mple\Desktop\djangoGIS\gis\dev\_static\vihlop2\vihod.shx']

zip\_subdir = "somefiles2"

zip\_filename = "%s.zip" % zip\_subdir

s = StringIO.StringIO()

zf = zipfile.ZipFile(s, "w")

for fpath in filenames2:

# Calculate path for file in zip

fdir, fname = os.path.split(fpath)

zip\_path = os.path.join(zip\_subdir, fname)

# Add file, at correct path

zf.write(fpath, zip\_path)

zf.close()

# Grab ZIP file from in-memory, make response with correct MIME-type

resp2 = HttpResponse(s.getvalue(), content\_type="application/x-zip-compressed")

# ..and correct content-disposition

resp2['Content-Disposition'] = 'attachment; filename=%s' % zip\_filename

return resp2

Додаток 2

Код файлу models.py

from \_\_future\_\_ import unicode\_literals

from django.contrib.auth.models import User

from django.db import models

SHORT\_TEXT\_LEN=700

# Create your models here.

class Articles(models.Model):

title = models.CharField(max\_length=200)

text = models.TextField()

user = models.ForeignKey(User)

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

return self.title

def get\_short\_text(self):

if len(self.text)>SHORT\_TEXT\_LEN:

return self.text[:SHORT\_TEXT\_LEN]

else:

return self.text

class Points(models.Model):

X=models.DecimalField(max\_digits=19, decimal\_places=10)

Y=models.DecimalField(max\_digits=19, decimal\_places=10)

type=models.CharField(max\_length=200)

Додаток 3

Код файлу admin.py

from django.contrib import admin

from gis.models import Articles, Points

admin.site.register(Articles)

admin.site.register(Points)

Додаток 4

Код файлу urls.py

from django.conf.urls import url, include

from django.contrib import admin

import gis.views

urlpatterns = [

url(r'^$', gis.views.home, name='home'),

url(r'^about/$', gis.views.about, name='about'),

url(r'^articles/(?P<article\_id>[0-9]+)/$', gis.views.show\_articles, name='article'),

url(r'^map/$', gis.views.mymap, name='map'),

url(r'^search-form/$', gis.views.search\_form, name='searchform'),

url(r'^search/$', gis.views.search, name='search'),

url(r'^search2/$', gis.views.search2, name='search2')

]

Додаток 4

Частина коду файлу test.html

<p>Set value 1 if you want to download temporary water areas or 2 to load permanent water areas</p>

<form action="/search/" method="get">

<input type="text" name="q">

<input type="submit" value="Download">

</form>

<p>Ocherednoy rofl tolko s BD</p>

<form action="/search2/" method="get">

<input type="text" name="w">

<input type="submit" value="Download">

</form>

Додаток 5

Код файлу заповнення таблиці БД

import arcpy

import sqlite3

conn = sqlite3.connect(r'C:\Users\S1mple\Desktop\djangoGIS\db.sqlite3')

cur = conn.cursor()

for row in arcpy.da.SearchCursor(r'C:\Users\S1mple\Desktop\test\points.shp', ["SHAPE@XY","type"]):

x, y = row[0]

z=row[1]

cur.execute('''INSERT INTO gis\_points (X, Y, type)

VALUES ( ?, ?, ? )''', ( x, y, z, ) )

conn.commit()

print('Done!')

cur.close()