Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка веб застосунку звикористанням фреймворку Flask та графічної бібліотеки Folium

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
старший викладач Жежерун
О.П

_____
(підпис)
“____” _____ 2020 р.

Виконав студент 4-го курсу
Стахурський Д.В.
“____” _____ 2020 р.

Київ 2019

Календарний план виконання курсової роботи

Тема: Використання загорткових нейронних мереж для розпізнавання тексту

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	Жовтень-листопад 2019р.	
2.	Огляд літератури за темою роботи	Листопад-грудень 2019р.	
3.	Аналіз фреймворку	Січень 2020р.	
4.	Розробка веб - застосунку	Лютий 2020р.	
5.	Реалізація відображення даних	Березень 2020р.	
6.	Реалізація графічного інтерфейсу застосунку та тестування	Квітень 2020р.	
7.	Написання текстової частини.	Квітень 2020р.	
8.	Перегляд курсової роботи науковим керівником	Квітень 2020р.	
9.	Створення презентації	18.04.2020	
10.	Захист курсової роботи	24.04.2020	

Студент Стахурський Д.В. _____

Керівник Жежерун О.П. _____

“_____”_____ 2020 р.

Зміст

Вступ

Робота присвячена розробці веб – застосування з можливістю візуалізації даних. В дані роботі описана розробка веб – застосунку з використанням мікрофреймворку Flask та графічної бібліотеки Folium. Мова, використана для розробки застосунку - Python

Розділ 1. Історичні відомості

Веб сайти, здебільшого це сукупність файлів, які зберігаються на комп’ютері, який називаєтьмя сервером. В разі коли сервер підлючений до інтернету, ви можете завантажити даний веб-сайт використовуючи браузер (наприклад, Chrome, Firefox або Safari). Кожен раз перебуваючи в інтернеті. Браузер в свою чергу це інструмент, який надає змогу переглянути те, що створив веб-розробник. В приклад можна навести сайти, починаючи з невеликих веб-сторінок предназначених для малого бізнесу і закінчуючи велечезними і складними веб-додатками, таких як Amazon, Twitter, Facebook.

Проте, з чого ж все почалося. До 1985 року глобальний Інтернет почав поширюватися в Європі, і почала створюватися система доменних імен (на якій будується Єдиний локатор ресурсів). У 1988 році було встановлено перший прямий IP-зв'язок між Європою та Північною Америкою, і було розпочате відкрите обговорення можливості веб-подібної системи в CERN. Інтенет, як цивільна структура фактично розпочав свій шлях в 1989 році. Одними з його безпосередніх засновників можна вважати Тіма Бернерса-Лі та Роберта Кайлі, які розробили паттрен на основі гіппертексту (HTTP + HTML = WEB). Колеги опублікували більш офіційну пропозицію щодо створення "проекту гіпертотексту" під назвою "WorldWideWeb" (одним словом) як "павутини" "гіпертекстових документів – 12 листопада 1990 року. Дана система була предназначена для полекшення і пришвидшення обміном інформації для Європейської організації ядерних досліджень. Це було реалізовано через мережу посилань, або гіперпосилань між різними документами, які могли б візуалізовуватися за допомогою конкретної програми, а саме браузера. Саме тому Web1.0 був задуманий, який сукупність пов’язаних статичних документів. Це надало змогу проводити швидші, а головне дистанційні дослідження. Зі зростанням популярності системи також зростала і її продуктивність, та коло зацікавлених. Тодішня рання веб-спільнота розробила декілька революційних ідей, які стимулювали поширенню інтернета і навіть сьогодні є неманш популярними далеко за межами технологічного сектору:

- Децентралізація передбачала собою відсутність центрального органу керування, що нівелювало, будь яку, цензуру та контрольований нагляд за інформацією розміщеною на просторах інтернету.

- Недискримінація визначала підхід, коли користувач, який платить за підключення до Інтернету з певною якістю послуг, а інший за ідентичну, або більшу якість обслуговування, то вони можуть спілкуватися на одному рівні Цей принцип справедливості також відомий як чистий нейтралітет.

- Дизайн знизу: Замість написанням коду, невеликою групою експертів та їхнього безпосереднього контролю, цей підхід перебачав залучення і заохочення усіх бажаючих, взяти участь в експеременті

- Універсальнісь: щоб в кожного була можливісь опублікувати, щось в інтернеті, усі комп'ютери повинні підпорядковуватися спільним падтерном взщаємодії, один з одним, незважачи на те, яке обладнання використовують користувачі, в якій частині світу вони знаходяться, або яким культурним і політичним переконанням вони підпорядковуються

- Консенсус: Для реалізації універсальних стандартів, усі, без виключення, повинні погодитися з їхнім використанням.

Згодом були розроблені методи, які дозволяли створювати сторінки з динамінчним контентом, що породило Web 1.5 у 1997 році. Відмінністю динамічних сторінок було те, що вони генерувалися на основі параметризованих запитів. Тоді і з'явилися CGI програми, які виконуватися на сервері і мали змогу отримувати параметри від клієнтів. Незвачаючи на свою іноваційність ці прогами надто перевантажували сервер, що в свою чергу стимулювало винекнення модульних систем, котрі були більше інтегровані в сервер, а також інтерпретованих і більш гнучних мов програмування, які дозволяли включати код в саму HTML – сторінку.

Всесвітня павутина відрізняла рядом відмінностей присутніх іншим гіпертекстовим системам, доступний на той час. Мережа передбачала лише однонаправлені посилання, що в свою чергу дозволяло посилатися на сторонній ресурс без безпосереднього підтвердення власника ресурсу. Дане нововвдення в значній мірі зменшило труднощі з реалізацією браузерів та серверної частини. На відміну від попередників таких як HyperCard, за рахунок децентрилізації всесвітньої павутини, всі охочі отримали змогу самостійно розробляти і модернізувати серверну частину та додавати розширенням без ліцензійних обмежень. 30 квітня 1993 року, CERN оголосили, що всесвітня павутина може стати безкоштовною. На думку

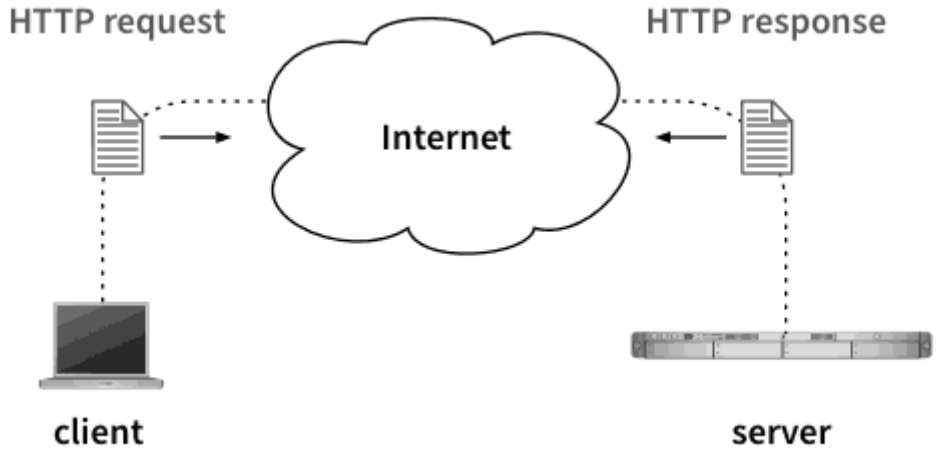
істориків переломний момент в розповсюдженні і зростанні інтернету відбувся в 1993, а саме, за рахунок, впроваження графічного браузера – Mosaic. Даний браузер розроблений в Національному центрі додатків для обчислювальної техніки в Університеті Іллінойсу в Урбані – Шампейн (NCSA-UIUC) . Розробкою керував Марк Андрессен, тоді як фінансування надійшло від Ініціативи високоефективних обчислень та комунікацій США та Закону про обчислювальну технологію високої продуктивності 1991 року. До випуску Mosaik, ніхто особливо не асоціював взаємодію графіки та текстом на веб-сторінках. Інтернет, як такий на той момент був менш популярний як старіші протоколи, такі як Gopher та широкоформатні інформаційні сервери (WAIS).

В жовтні 1994 року, Тім Бернерс-Лі покинув Європейську організацію ядерних досліджень (CERN) та став засновником Консорціуму Всесвітньої павутини (W3C). Даний консорціум був реалізований на базі Массачусетського технологічного інституту з лабораторії комп'ютерних наук (MIT / LCS) при підтримці Агенції прогресивних дослідницьких проектів в галузі оборони (DARPA), яка стала першопрохідцем Інтернету. В 1995 році, по іншу сторону океану на базі Французьох національної лабораторії комп'ютерних досліджень(INRIA), був заснований другий сайт за підтримки Генеральної дирекції Європейської Комісії, практично в той самий час в Японії, а саме в університеті Кейо було створено третю континентальну платформу. Дана тендеція спонукала розвиток і становлення великої кількісті сайтів, що згодом переросли в потужні сервіси, всесвітньовідомі і нас ьогоднішній день

Підключення сайтів до мережі і стовренні сотні нових стимулювало і спонукало розробку міжнародних стардантів для протоколів та форматування. Також активно відбувалася розробка веб-стандартів, таких як мова розмітки для побудови веб-сторінок. Всесвітня павутина дозволила поширити інформацію через Інтернет за допомогою простого у користуванні та гнучкому форматі. Таким чином, це досить вагомо вплинуло на популяризацію використання Інтернету. Хоча обидва терміни іноді поєднуються у популярному використанні, всесвітня павутина не є синонімом Інтернету. Мережа - це інформаційний простір, що містить гіперпосилання з документами та іншими ресурсами, ідентифіковані їх URI. Інтернет реалізований як клієнтське, так і серверне програмне забезпечення за допомогою Інтернет-протоколів, таких як TCP / IP і HTTP.

HTTP

HTTP – дуже широко розповсюджений протокол передачі даних, який попередньо був презначений для передачі гіпертекстових докуметів. Власне абрівіатура розшифровується як протокол передачі гіпертексту. Відповідно до спецмфікації OSI, HTTP відноситься до протоколів прикладного(верхнього, 7-го) рівня. Даний протокол передбачає використання клієнт-серверної структури передачі данних. На стороні клієнта формується запит і відправляється на сервер, після того як сервер приймає запит і успішно його опрацьовує, він одномоментно формує відповідь і повертає її зворотньо, на сторону клієнта. Після цього клієнський застосунок може надіслати інший запит, і процес “спілкування” відбудеться по аналогічному шляху.



На сьогоднішній день саме завдяки протоколу HTTP відбувається взаємодія всесвітньої павутини. Варто зазначити, що даний протокол часто застосовується при передачі даних іншими протоколами, а саме протоклами прикладного рівня, таких як SOAP, XML-RPC та WebDAV. В даному випадку прийнято говорити, що протокол HTTP використовується як “ траспорт ”. API, а також багато інших програмних продуктів передбачає використання HTTP, для передачі інформації. Дані в такому випадку можуть мати будь – який формат, наприклад, XML або JSON.

В основному, передача даних здійснюється через TCP/IP з’єднання. Дане програмне забезпечення використовує TCP-порт 80, вказаний порт зазвичай використовується зі сторони клієнта по замовчуванню, проте при необхідності він може бути змінений на будь-який інший.

Ресурси і методи

Для відправлення HTTP – запиту, необхідно сформувати пошукову строку,в якій в свою чергу як мінумум потрібно задати один заголовок, а саме Host. Даний заголовок є обов'язковим і повинен бути присутнім в кожному запиті. Варто зазначити, що перевизначення доменного імені відбувається на стороні клієнта, і відповідно коли ми відкриваємо TCP-з'єднання, то завантажений сервер не має ніякої інформації про цей, який саме адреса використовується для з'єднання.

```
GET /index.php HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5
Accept: text/html
Connection: close
```

Приклад HTTP запиту

Розглядаючи HTTP запит можна виокремити такі його частини:

- **Метод**: Дану складову запиту можна інтерпретувати як послідовність символів, окрім розділювачів і службових знаків, і визначає операцію, яку необхідно виконати. Розглядаючи специфікацію HTTP 1.1 можна зазначити, що існує необмежена кількість методів, які можна використати, проте переважно використовуються стандартні методи такі як:
    1. GET – отрмання даних
    2. POST – надсилання даних
    3. PUT – вставка, обновлення даних
    4. DELETE – видалення даних

- **URI** (*Uniform Resource Identifier)* - шлях до конкретного ресурсу, над яким необхідно здійснити певну операцію, наприклад виористання метода GET. Деякі запити можуть не відноситися до конкретного ресурсу, в такому випадку на місці URL, може знаходитися службовий символ "*". Прикладом цьому може послугувати запит, який безпосередньо адресується серверу, а не якомусь конкретному ресурсу

- **Заголовки** – це набір пар,ім'я та значення. В заголовках передається різномантіна інформація про службу, яка може містити тип кудування повідомлень, назву, а також версію браузера.

- **Тіло повідомлення** – власне, самі дані, які ми хочемо передати. Дані можуть варіюватися від html – сторінки, яку повертає сервер, до фотографій, які користувач, власне загружає в зворотньому напрямку

Відповідь серверу

Відповідь сервера має наступну структуру:

```
HTTP/1.1 302 Moved Temporarily
Server: nginx
Date: Sat, 08 Mar 2014 22:29:53 GMT
Content-Type: text/html
Content-Length: 154
Connection: keep-alive
Keep-Alive: timeout=25
```

Код статусу – три цифри котрі визначають статус і результат поверненого запиту. Наприклад, коли ми використали запит GET і сервер успішно опрцював наш запит і повернув очікувану інформацію, код статусу в такому випадку – 200. В разі, коли сервер повідомляє, що такого ресурсу не існує – це код 404. В разі, якщо у клієнта недостаньо привілегій для отрмання того чи іншого ресурсу, код статусу – 403. Специфікація HTTP 1.1 визначає 40 різних кодів HTTP, а також допускається розширення протоколу і використання додаткових кодів станів.

Пояснення до коду стану (Reason Phrase) – пояснення до коду відповіді призначене для спрощеного розуміння людиною і можливісю краще зрозуміти природу помилки. Дане пояснення може бути стандартним, або визначеним самими розробниками при створенні ресурсу.

Тіло відовіді: Для визначення закінчення використовується значення заголовка Content-Length (в даному випадку відповідь містить 7 вісімкових байтів: слово «Wisdom» і символ розриву рядків).

Безпека HTTP

Сам по собі протокол HTTP не передбачає використання шифрування для передачі інформації. Проте для даного протоколу є досить популярним

розширення, яке реалізує передачу інформації, за допомогою криптографічного протоколу SSL або TLS.

Дане розширення - HTTPS (HyperText Transfer Protocol Secure). Для конкретних з’єднань здебльшого прийгято використовувати TCP-порт 443. HTTPS передбачає захист даних від перехоплення, а також гарантує захист від можливих атак типу - man-in-the-middle, в ситуації, коли сертифікат верифікується на стороні клієнта, і при цьому приватний ключ сертифіката, не є скомпрометованим, користувач не використання непідписаного сертифіката, і на комп'ютері користувача не були впроваджені сертифікати центру сертифікації зловмисника.

Додаткові можливості

Протокол HTTP володіє великою кількістю додаткових можливостей, прикладом цього може послугувати специфікація HTTP 1.1, яка надає змогу при заміні заголовку Upgrade, використовувати інший протокол для передачі даних. Якщо серверу потрібно зробити перехід на обмін даними по іншому протоколу, то він може повернути клієнту відповідь зі статусом «426 Upgrade Required», і в цьому випадку клієнт може відправити новий запит, вже з заголовком Upgrade.

Така можливість використовується, зокрема, для організації обміну даними по протоколу WebSocket (протокол, описаний в специфікації RFC 6455, що дозволяє обом сторонам передавати дані в потрібний момент, без відправки додаткових HTTP-запитів).

HTML

Мова для розмітки гіпертексту (HTML) – мова розмітки предназначена для розробки веб-сайтів та різноманітних додатків. HTML є нащадком SGML, яка в свою чергу була надто складна для пересічних людей. Виокремивши також каскадні таблиці стилів(CSS) та мову програмування JavaScript, можна зазаначити, що дане тріо формує фундамент всесвітньої павутини.

Одними з вагомих переваг HTML можна відмітити просуту, яка була досягнута за рахунок використання стуктрурних елементів, так званих дескрипторів, або ж тегів. Також це можливість форматування документа без посилання на засоби відображення. Веб-браузери отримують HTML – документи з веб – сервера, або ж локального сховища вашого комп'ютера. HTML описує структуру веб-сторінки семантично та спочатку містить підказки для зовнішнього вигляду документа.

Елементи HTML - це складові HTML-сторінок. За допомогою HTML-конструкцій, зображення та інші об'єкти, такі як інтерактивні форми, можуть вбудовуватися у візуалізовану сторінку. HTML забезпечує засіб для створення структурованих документів, позначаючи структурну семантику для тексту, таких як заголовки, абзаци, списки, посилання, цитати та інші елементи. Елементи HTML розмежовані тегами, написаними за допомогою кутових дужок. Такі теги, як <img /> та <input /> безпосередньо вводять вміст на сторінку. Інші теги, такі як <p> оточують і надають інформацію про текст документа, і можуть включати інші теги як під-елементи. Браузери не відображають теги HTML, але використовують їх для інтерпретації вмісту сторінки.

Префікс WWW

Протягом багатьох років, назви хостів, які використовуються в всевітній павутині, починаються з www. Варто зазначити, що ім’я веб-сервера може бути довільним і почнатися наприклад з ftp для FTP-сервера. Дані імена відображаються як імена доменних імен. Використання префікса www непідпорядковується жодним технічним, або політичним стандартам. За словами працівників Європейського центру ядерних досліджень, використання www, як субдомені відбулося чисто випадково. Сторінка проекту World Wide Web повинна була бути опублікована на веб-сайті www.cern.ch, тоді як info.cern.ch повинен був бути домашньою сторінкою CERN, проте записи DNS ніколи не перемикалися, і практика попереднього розміщення www на веб-сайті установи згодом доменне ім’я було скопійовано.

Використання даного піддомену на сьогоднішній день є також досить популярним. Дуже часто при введені користувачем незвершеного доменного ім’я, деякі веб-браузери автоматично намагаються додати префікс "www" до початку його, можливо, ".com", ".org" та ".net "наприкінці, залежно від того, що може бути відсутнім. Наприклад, введення "microsoft" може бути перетворено на http://www.microsoft.com/, а "openoffice" на http://www.openoffice.org. Ця функція почала з'являтися в ранніх версіях Firefox.

Піддомен "www" можна налаштувати за допомогою запису DNS CNAME, який можна оновлювати та змінювати миттєво. Користувачі повинні бути обережні щодо зберігання файлів cookie клієнтів. Файл cookie, sessionStorage або localStorage, встановлені для домену, який не є www, діляться всіма піддоменами. Якщо основний веб-сайт - mysite.com - задає 10Kb даних cookie, він буде переданий з кожним запитом та відповіддю для app.mysite.com, використовує ці програма ці дані чи ні. Варто зазанчити, що префікс www, є важливим для таких застосунків, як клієнти електронних пошт, а також текстові процесори, які перетворють текст у посилання. Незважаючи на технічні проблеми, використання домену, який не є www, рідко є проблемичним, у випадку відсутності необідності щодо хостингу чи застосування. Кореневі адреси легше читати, швидше набирати та легше вміщувати у менших (мобільних) інтерфейсах браузера.

Роль сервера в мережі

Будь-який комп’ютер, на якому встановлене необхідне прогамне забезпечення, може функціонувати як сервер. В сучасному світі сереверами прийнято вважати надпотужні машини, які обслуговують веб-платформи розраховані на міліони користувачів. В основному веб-сервер може містити одни або деклька веб-сайтів. Веб-сервер обробляє вхідні мережеві запити через HTTP та кілька інших пов'язаних протоколів. Сервером також можна назватипрогамне забезпечення, яке зконцентроване на певному завданні. Однак потужне обладнання, яке підтримує це програмне забезпечення, також називається сервером. Це тому, що серверне програмне забезпечення, яке координує мережу сотень чи тисяч клієнтів, вимагає апаратного забезпечення, більш надійного, ніж комп'ютери для використання споживачами.

Види серверів

В залежності від складності і затратності того, які сервер виконує функції, варифікації клієнтів, або баз даних товарів, деякі реаліхації використовують один сервер для кількох цілей. Мережа, орінтована на середню компанію, може використовувати декілька типів серверів, наприкад:

- **Веб – сервер**: Відповідає за відображення сторінок та запуск програм через веб-браузери. Сервер, котрий взаємодіє з вашим браузером, рендерить текст сторінки, зображення та можливі її інтерактивні компоненти. В даному випадку клієнтською програмою є веб-переглядач, такий як Internet Explorer, Chrome, Firefox, Opera або Safari. До функціоналу веб-сервера можна віднести виконная багатьох задач, таких як завантаження, а також резервне копіювання файлів в інтернеті через хмарні служби зберігання даних, або ж сервіси резервного копіювання, на випадок непередбачуваних ситуацій.

- **Сервер електронної пошти:** Дані сервери надсилають та отримуютьповідомлення електронної пошти. Програмне забезпечення відповідає за підлючення до таких серверів, як IMAP, або POP для завантаження повідомлення на пристій користувача. Сервер SMTP, в свою чергу відповідає за надистання елктронних листів.

- **FTP-сервер:** Завдання даного сервера полягає в переміщенні файлів, використовуючи інструменти протоколу передачі файлів. Доступ до FTP – сервера можна здійснити віддалено за допомгою клієнтских програм, які мають спільний доступ до файлів, розташованих на сервері, або ж за використанням спеціального прогамного забезпечення сервера FTP**.**

- **Identity сервер:** Реалізує варифікацію користувачів, перевірку їхніх даних, а також відповідає за рівні безпеки відповідні до повноважень коритсувачів.

На сьоголнішній день існують сотні спеціалізованих серверів, котрі підтримують комп'ютерні мережі. Окрім корпоративних випадків, пересічні користувачі досить часто взаємодіють з ігровими серверами, серверами, які реалізують онлайн чати, а аткож різномантіними стрімінговими серверами. Деякі сервери зосередженні у вузькоспеціалізованих напрямках, прикладом цьому можуть послужити DNS, або proxy сервери.

Типи мережевих серверів

В переважній більшості у всесвітній павутині розповсюджена мережева модель клієнт – сервер, котра інтегрує веб-сайти та комунікаційні послуги. Існує альтернативна модель, котра називається одноранговою мережею, і принцип її дії полягає в можливості всім пристроям функціонувати як сервер, або клієнт за потребою. Peer networks здатні реалізувати більшу конфеденційність, оскільки з'єднання між комп'ютерами є більш вузьконаціленими. Однак внаслідок, обмеження пропускної здатності, домінуюча більшість однорангових систем недостатньо надійні, щоб витримувати стрибки трафіку.

Серверні кластери

Кластер як термін використовується для реалізації спільних обчислювальних ресурсів. Здебільшого кластер відповідає за інтеграцію та взаємодію, двох, або більше спільних, обчислювальних ресурсів. Зазвичай кластер інтегрує ресурси двох, або більше обчислювальних пристроїв, які при необхідності можуть функціонувати окремо для реалізації якоїсь конкретної мети(часто робочої станції, або серверного пристрою).

Ферма веб-серверів – це сукупність мережевих веб – серверів, котрі є взаємо пов’язані, та мають доступ до певного веб – сайту. Варто зазаначити, що дані сервери функціонують як кластери концептуально. Деякі спеціалісти вважають, що клсифікація даних ферм залежить від конфігурацій апаратних та програмних засобів.

Сервери в домашніх умовах

Оскільки сервери – це програмне забезпечення, ентузіасти можуть реалізовувати сервери в домашніх умовах, котрі будуть доступні, для пристроїв, підключених до локальної мережі, або ж для пристроїв котрі знаходяться поза цією мережею. В приклад можна навести жості диски, котрі використовують принцип мережевого сховища, в домашній мережі, щоб отримати доступ до спільного набору файлів.

Програмне забезпечення мультимедійного сервера Plex надає змогу користувачам переглядати цифрові медіа-файли на незалежних пристоях, незважаючи на те, розіміщені дані у хмарі, або ж на локальномсу ПК.

В сучасних реаліях, тривалість роботи є критично важливою для більшості серверів і непередбачуване відключення може нанести значних компанії. Дані про помилку, виникшу на сервері можна розпізнати за домогою стандартного коду статусу HTTP. Коли веб-сервер видаляє інформацію назавжди, або тимчасово, користувачі зможуть отримати доступ до даних файлів, якщо відбулася попередня архівація. Wayback Machine - один із прикладів реалізації веб-архіватора, який зберігає інформацію з веб-сторінок і файлів, що зберігаються на веб-серверах.

Великі установи, що мають декілька серверів, в більшості випадків не мають безпосереднього доступу до цих серверів, а лише віддалено. Ці сервери також іноді є віртуальними машинами, тобто один запам'ятовуючий пристрій може розміщувати декілька серверів, що економить фізичний простір та гроші.

Python

Становлення мови прогамування python розпочалося на прикінці 1980-років. Дана мова була задумана в кінці 1980-х і її реалізація була розпочата в грудні 1989 року, нідерландським дослідником - Гвідо ван Россумом. Python є нащадком мови прогамування ABC, з можливістю обробки виключень і помилок, а також взаємодія з операційною системою Amoeba. Напрямок розвитку Python, власне можна відслідкувати з його назви, яка була придумана громадою розробників. Дана мова, на сьогоднішній день широко використовується в багатьох галузях і являється мовою програмування високого рівня. Його дизайн та філософію підкреслює читабельність коду, а синтаксис дає змогу лаконічніше описувати поставлені задачі, чим це було б можливо використовуючи C++, або Java. Мова надає змогу реалізовувати конструкції для побудови чітких і зрозумілих прогам, як малого так і великого маштабів.

Python реалізує декілька парадигм прогамування, включаючи об'єктно-орієнтовану, імперативну та функціональну. Він має динамічну систему типів та прибиральника сміття. Однією із значних переваг Python є велика кількість відкритих і доступних бібліотек, практично в усіх галузях, котрі постійно постійно доповнюються і розиваються, за рахок величезної спільноти.

Інтерпретатори Python доступні для установки на багатьох операційних системах, що дозволяє використовувати написані на ньому програми, в широкому спектрі систем. В Python можна використовувати і інші парадигми, таких як design by contract та логічне програмування, за допомогою ззовнішніх розширень. Python використовує динамічне введення тексту та циклічний пошук прибиральника сміття для управління пам’яттю. Також однією з особливостей Python є пізнє зв’язування, котре пов’язує імена методів та змінних під час виконання програми. Дизайн Python надає змогу розробляти програми прогами в функціональному стилі, дотримуючись традицій Lisp.

Мова має вбудовані функції фільтрації, роботи з кортежами, списками, генераторами випадквоих значень. Стандартна бібліотека має два модулі itertools та functools. Також Python має значні переваги перед різними мовами програмування, наприклад:

- Чистий синтаксис, дозволяє розбивати програму на окремі блоки та модулі

- Довільний стиль написання прогами (що характерно для більшості інтерпретованих мов)

- Принцип роздільного створення модулів передбачає змогу викорасання тільки необхідних елеметів і мінімальну кількість написаного коду

- використання Python в інтерактивному режимі (дуже корисно для експериментів та вирішення простих проблем)

- наявність великої кількості бібліотек для візуалізації гафічного інтерфейсу і даних

- підходить для розв'язання математичних задач (реалізація операцій з комплексними числами)

- з цілими числами довільної величини, командний рядок може використовуватися як потужний калькулятор).

Однак у Python все ж є деякі недоліки. Python, як і в багатьох інших інтерпретованих мовах прогамування, де не застосовуються, JIT-компілятори мають загальний недолік - відносно низьку швидкість виконування програми. Крім того, відсутність статичного набору тексту та деякі інші фактори, на жаль, не дозволяють, під час компіляції, реалізувати механізм перевантаження функцій.

Окрім того, що Python це добре бродумана і збалансована мова програмування, його активно використовують для реалізації в найрізномантніших областях. Це може бути створення сценаріїв різних компонентів та реалізації автономних програм. Як мова загального призначення, python розвивається в багатьох сферах від розробки веб-сайтів та ігор, до робототехніки та управління космічними кораблями. Програми Python можуть шукати файли та дерева каталогів, запускати інші програми, робити паралельну обробку процесів та ниток. Стандартна бібліотека Python оснащена прив'язками POSIX, розширення імен файлів, утиліти zip-файлів, аналізатори XML та JSON, обробники файлів CSV та інше. Крім того, основна частина системних інтерфейсів Python адапотовані для інтеграції; наприклад, сценарій, який зазвичай копіює дерева каталогів працює без змін на всіх основних платформах Python.

Python постачається зі стандартним об'єктно-орієнтованим інтерфейс API Tk GUI під назвою tkinter (Tkinter в 2.X), що дозволяє програмам Python реалізовувати портативні графічні інтерфейси з авторською реалізацією. Графічні інтерфейси Python / tkinter працюють без змін у Microsoft Windows, Linux та Mac OS. Крім того, пропонує графічний інтерфейс API wxPython, розроблений на основі бібліотек C ++

Набори інструментів вищого рівня, такі як Dabo, побудовані на основі базових API, таких як wxPython та tkinter. За допомогою відповідної бібліотеки ви також можете використовувати підтримку GUI в інших наборах інструментів у Python, такі як Qt з PyQt, GTK з PyGTK, MFC з PyWin32, .NET з IronPython та Swing з Jython або JPype.

Інтернет-сценарії Python

Python оснащений інтернет – модулями, котрі дозволяють реалізовувати обширний спектр мережевих завдань та побудуві клієнт-серверної архітектури. Сценарії мають змогу взаємодіяти за рахунок сокетів, оброляти та надсилати різноманітні дані з форм та передавати їх на сервер, здійснювати передачу файлів викорстовуючи FTP, парсити, а також генерувати XML, або JSON документи. Також можна аналізувати електронну пошту, відображати веб-сторінки за URL-адресами, аналізувати та парсити HTML заданих веб-сторінок; підтримувати зв’язок використовуючи XML-RPC, SOAP та Telnet.

Крім цього доступний широкий спектр сторонніх інструментів для веб – розробки в python. Також варто відмітити, систему HTMLGen, котра здійснює генерацію HTML. Пакет mod_python ефективно працює з Python на веб-сервері Apache та реалізує шаблони на стороні сервера. Фреймворк Jython забезпечує надійну інтеграцію Python / Java та підтримує кодування серверних аплетів, котрі виконуються на стороні клієнта.

Також варто зазаначити повномаштабні фреймворки для веб-розробки, такі як як Django, TurboGears, web2py, Pylons, Zope та WebWare. Дані фреймворки втілюють швидку побудову повнофункціональних та високоефективних веб-сайтів із Python. Багато з них включають такі функції, як об'єктно-реляційні картографи, архітектуру Model/View/Controller, сценарії та різномантні шаблони на стороні сервера та підтримка AJAX, щоб забезпечити повноцінні рішення для веб-розробки на рівні підприємства.

Візуалізація даних в Python

Візуалізація даних – напрямок, що займається дослідженням та знаходженням тенденцій, кореляцій даних, за рахунок перегляду даних в візуальному контексті. За допомогою інтерактивної візуалізації можна змоделювати концепцію на крок далі, використовуючи технологію для деталізації деталей у графіках та графах, або ж картах, інтерактивно змінюючи дані, які ви бачите та як вони обробляються.

Концепція використання зображень для кращого розуміння даних існувала століттями - від карт та графіків у 17 столітті до винаходу кругової діаграми на початку 1800-х років. Через кілька десятиліть один з найбільш відомих прикладів статистичної графіки стався, коли Чарльз Мінард склав карту Наполеонова вторгнення в Росію. На карті було зображено чисельність армії, а також шлях відступу Наполеона з Москви, також на даній карті була зображена статистика, стосовно температури повітря на період вторгнення та часових відрізків, для більш глибокого розуміння події.

Проте, дана галузь стала посправжньому доцільною та багатообіцяючою в період технологічного буму і появлення потужних комп'ютерів, котрі дали можливість обробляти велику кількість даних на блискавичних швидкостях. Сьогодні візуалізація даних стала швидко розвивається поєднанням науки та мистецтва, що, безумовно, змінить корпоративний ландшафт протягом наступних кількох років.

Типи категорій візуалізації великих даних:

- **Temporal**: Візуалізацію даних можна віднести до часової категорії, у випадку, якщо задовільняються дві умови, а саме лінійнсть і одновимірність даних. Даний тип візуалізації зазвичай містить лінії, які або розташовані окремо, або перетинаються між собою, відповідно до часу початку та закінчення дослідження.

- **Hierarchical:** Цей тип можна охарактеризувати як процес впорядкування груп всередені більших груп. Ієрархічні візуалізації найкраще підходять, у випадку відображення кластерів інформації, особливо якщо вони надходять з однієї точки. Недоліком цих графіків є те, що вони, як правило, складніші і важчі для читання, саме тому діаграма дерева використовується найчастіше.

- **Network**: Набори даних пов’язуються з іншими наборами даних. Візуалізації мережевих даних показують, як ці дані співвідносяться один до одного в мережі. Іншими словами, демонстрація зв’язків між наборами даних.

- **Multidimensional:** Так само, як і назва, багатовимірні візуалізації даних мають декілька вимірів. Це означає, що у наборі даних, завжди є 2 або більше змінних для створення візуалізації 3D даних. Через безліч одночасних шарів і наборів даних, ці види візуалізації, як правило, є найбільш яскравими або захоплюючими

- **Geospatial:** Візуалізації геопросторових, або просторових даних відображають реальні локації, накладаючи на географічні карти, дані які нас цікавлять. Даний тип візуалізації зазавичай використовується для відображення: демографічних, економічних, геополітичних та політичних даних.

Python пропонує безліч високорівневих графічних бібліотек, які містять в собі велику кількість різноманітних функцій, для створення інтерактивних, відображень.

До часто використованих і широпощирених графічних бібіліотек можна віднести:

- **Matplotlib**: Низькорівнева бібіліотека, одна з головних переваг полягає в можливості довільної реалізації. Дана бібліотека є найпопулярнішою бібліотекою, низького рівня з інтерфейсом, схожим на Matlab, яка пропонує, можливість власного вибору реалізації, ціною необхідності писати більше коду. Діаграми Matplotlib складаються з двох основних компонентів, осей (ліній, що розмежовують область діаграми) та фігури (осі, заголовки та речі, що виходять із зони осей). Бібліотека дає змогу реалізувати декілька графіків, однією фігурою. Це доволі зручно використовувати для порівняння діаграм, або для обміну даними між діагамами

- **Pandas Visualization:** Проста у використанні, високорівнева бібліотека, яка знаходиться у відкритому доступі. Однією з переваг цієї бібіліотеки

є вбудовані структри даних, такі як датафрейми та методи для аналізу даних. Бібілотека також має API більш високого рівня, порівнюючи з Matplotlib, і тому для реалізації схожого графіка, потрібно менше коду. Головним профілем Pandas є генерація інтерактивних звітів з даними користувачів, також є можливість переглядати розподіл даних, типи даних, можливі проблеми. Pandas можна встановити, використовуючи або pip, або conda.

- **Seaborn:** Seaborn - бібліотека, розроблена на основі Matplotlib. В основному, різницю можна відзначити, за рахунок, простішої реалізації та кращої візуалізаціїї графіків, функції для створення складних типів графіки лише одним рядком коду. При роботі з цією бібіліотекою необхідно ініціалізувати стиль графіки за допомогою sns.set (), без цієї команди графіка все одно матиме той же стиль, що і Matplotlib. Однією з найпопулярніших графічних реалізацій, що надаються Seaborn, є теплова карта. Вона дуже часто використовується для показу всіх кореляцій між змінними в дата-сеті. Ще однією популярною фінкцією є pairplot, котра показує зв'язки між усіма змінними, відповідно до збільшення розміру даних, час обробки росте експотенціально.

- **Bokeh:** Основна концепція даного фреймворку полягає в покрокововій візуалізації графіку, або фігури. Першочергово відбувається створення фігури, після чого до фігури додаються елементи так звані гліфи. Гліфи можуть набувати багатьох форм в залежності від необхідних вимог, це можуть бути кола, лінії, дуги

Розділ 2. Теоретичні відомості про Flask та Folium

2.1 Розробка додатків з використанням фреймворку Flask

Flask це ліцензований BSD мікрофреймворк на осонові Werkzeug та Jinja2. Особливість мікрофреймворків, полягає в тому, що вони намагаються надати розробнику тільки необхідні компоненти для реалізації поставленої задачі. Мікрофрейморки можуть бути спеціально розроблені для створення APIs для певного сервісу, або сайту. Flask доволі простий, але одночасно і дуже гнучкий, що дає можливість розробникам використовувати тільки необхідні конфігурації, що полекшує розробку програм, або плагінів. Flask був розроблений Росоо, та одразу опинився у відкритому доступі. На сьогоднішній день фреймворк підтримується проектом The Pallets Project, котрий займається розробкою нових та модернізацією старих компонентів. Варто зазначити, що у Flask, дуже активна спільнота розробників та декілька великих форумів.Розгортання та моніторинг API є важливою частиною REST API, в Flask в залежності від маштабування API змінюється і сама парадигма розробки.

Два головні компоненти Flask це Werkzeug і Jinja2. Попри те, що Werkzeug несе відповідальність за надання маршрутизації, налагодження та інтерфейс шлюзу веб-сервера (WSGI), двигуном шаблону являється Jinja2. Сам по собі, Flask не підтримує доступ до бази даних, автентифікацію користувачів чи будь-яку іншу утиліту високого рівня, але він реалізує підтримку велкої кількості розширень, котрі реалізують вище перерахований функціонал. Простий додаток можна реалізувати навіть в одному файлі, проте при реалізації великого застосунку, краще розподілити програму на модулі. Модульна структура також один з переваг Flask. Сама ідея даного фреймворку полягає в реалізації надійної основи додатку, а функціонал верхнього рівня втілюють розширення.

Flask-спільнота досить велика та активно займається розробкою сотень розширень, які одразу публікуються у відкритий доступ. Команда Flask core постійно маніторить нові розширення та гарантує, що затверджені розширення сумісні з майбутніми випусками. Будучи мікрофреймворком, Flask забезпечує розробникам гнучкість у виборі дизайнерських рішень, котрі чудово

вписуються в архітекру проекту. Також розробники ведуть реєстр розширень, який регулярно оновлюється та постійно підтримується.

Flask, як і всі інші бібліотеки Python, можна встановити, використовуючи індекси пакетів Python (PPI), і його дуже просто налаштувати і почати розробляти.

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, From Flask!'

if __name__ == '__main__':
      app.run()
```

Даний код імпортує бібліотеку Flask, ініціює додаток, створивши екземпляр класу Flask, оголошує маршрут, а потім визначає функцію для виконання при виклику маршруту. Цього коду достатньо для запуску першої програми Flask. Цей код запускає дуже простий вбудований сервер, котрий чудово пыдходить для тестування, але недостатньо потужний для введеня додатку в експлуатацію.

Як уже обговорювалося, Flask не здійснює підтримку доступу до бази даних, і для здійснення взаємодії з БД, здебільшого використовують розширення Flask під назвою Flask-SQLAlchemy, що надає підтримку бібліотеки SQLAlchemy. По суті, SQLAlchemy - це набір інструментів Python SQL та Object Relational Mapper, що забезпечує розробникам повну потужність і гнучкість SQL.

SQLAlchemy здійснює повну підтримку прадигм дизайну на рівні підприємства та розроблена для високоефективного доступу до бази даних, зберігаючи ефективність та простоту використання. Хорошим тоном при розробці застосунку вважається, реалізація модуля аутентифікації користувача, CRUD (створення, читання, оновлення та видалення даних), API REST для створення, пошуку, маніпуляцій та видалення об'єктів. Також Flask надає змогу інтеграції утиліти Swagger для створення документації API, написання тестів та їхньої інтеграції. Для вузьконаправленого тестування функцій, прийнято використовувати pytest, який є повнофункціональним інструментом для тестування Python – застосунків. Pytest дозволяє легко розробляти тести, і все ж ця бібліотека достатньо масштабована для підтримки складних випадків використання. Postman, являється повноцінною платформою API REST, і надає інструменти інтеграції для кожного етапу життєвого циклу API, що робить розробку API простішою та надійнішою.

2.1 Використання графічної бібілотеки Folium в розробці додатків

Folium - це потужна бібліотека Python, яка реалізовує декілька типів карт Leaflet. Той факт, що результати Folium є інтерактивними, робить цю бібліотеку дуже ефективною при побудові інформаційної панелі.

Folium використовує механізм шаблонів Jinja2 Python для візуалізації кінцевих результатів, а Pandas – відповідає за прив'язку статистичних даних CSV. Реалізація починається з імпорту, а потім відбувається визначення даних джерела.

Folium реалізує міст між можливостями обробки даних Python та можливостями візуалізації інтерфейсу, які пропонує JavaScript. Зокрема, це дозволяє розробникам Python інтегрувати дані GeoJSON і TopoJSON з бібліотекою Leaflet, однією з найбагатших бібліотек, котру використовують на фронтенді, для створення інтерактивних карт. Перевага використання такої бібліотеки, як Folium, полягає в тому, що вона безперебійно обробляє переклад між структурами даних Python та компонентами JavaScript, HTML та CSS.

До мінусів цієї бібіліотеки можна віднести проблеми з відображенням карт, у випадку комбінації маркерів та спливаючих вікон, візуалізовуючи велику кількість елементів.

При рендерину карти Folium, необхідно створити об’єкт самої карти, встановивши поряд координати центру карти, рівень масштабування карти, базової плитки для нашого фону.

Розглядаючи бібліотеку Folium, варто зазаначити декілька речей:

- Карта створена за допомогою даної бібіліотеки визначаюється як folium.Map object. Ми можемо додати інші об’єкти, поверх першого, таким чином реалізуючи більш детальна відображення і можливість додвати нові об’єкти.

- Бібілотека дозволяє власноруч створювати, або вибирати шаблони карт, наприклад, з OpenStreetMap, MapBox

- Folium надає змогу вибирати різні проекції на карті. Доволі часто використовують сферичну проекцію Меркатора, особливо при візуалізації площі, порівняно невеликих розмірів.

Розглядаючи атрибути бібліотеки варто відмітити такий параметр як location, котрий задає точку фокусу на карті. Атрибут zoom_start дозволяє змінювати маштаб карти. Параметр control_scale, вимикає масштаб карти при певному, заданому рівні збільшення. Це те, що іноді може бути корисним для користувача, щоб отримати уявлення про масштаби географічної області, котру він переглядає.

HeatMap().

Бібліотека Folium реалізує досить цікаві методи, наприклад *HeatMap()*. Цей метод використовується для накладання теплової карти на попередгьо створений об’єкт карти.

ClickForMarker()

Додавання даного метода до створеного об’єкта карти, надає можливість користувачу створювати маркери, на площині всієї карти

HeatMapWithTime()

Дозволяє анімувати карти, змінюючи параметри даних, що відображаються на карті. Для реалізації анімації необхідно створити список, що містить список значень, які ми хочемо відображати, згрупувавши їх за виміром, який ми хочемо використовувати.

2.3 Висновки до розділу 2

Провівши дослідження та ознайомившись з мійкрофреймвокром Flask, можна зробити висновок, що це чудове рішення для реалізації, як невеликих застосунків, розроблених в навчальних цілях, так і для розробки великих корпоративних проектів. Ключова відмінність Flask полягає в тому, що він розрахований на реалізацію надійної основи додатку, а функціонал верхнього рівня розроблюється з використанням зовнішніх розширень. Внаслідок того, що в даного фреймворка є велика спільнота розробників, він постійно вдосконалюється і розвивається.

Розглянувши графічну бібліотеку Folium, можна виділити багато хороших моментів, проте в ній пристуні і недоліки. Одним з ключових недоліків, є те що для даної бібліотеки є досить проблематично візуалізовувати карти з великою кількістю даних. Незважаючи на це, Folium є досить простою в реалізації, має велику кількість шаблонів, а також за рахунок своєї гнучкості, може реалізовувати складні структури.

Розділ 3. Опис реалізації прогамного продукту

3.1 Аналіз технічного завдання

Метою технічного завдання є проектування та розробка веб – застосунку “GreatMapper”. Суть даного застосунку полягає в візуалізації різноманітних географічних, політичних, геополітичних та екномічних даних. Також GreatMapper передбачає можливісь візуалізації даних користувачів.

Інтерфейс користувача реалізує:

- Реєстрацію
- Авторизацію
- Можливість завантажувати та візуалізовувати власні дані
- Можливість перегляду даних в зстосунку

В процесі проектування застосунку, було прийнято рішення використовувати реляційну базу даних. Також під час пошуку та збору матеріалів для розробки застонку, джерелом даних для візуалізації, було обрано веб – сайт, datasf.org.

На даному етапі, застосунок розміщений на локальній машині, проте у випадку розширення та модернізації розгорнути на хмарній платформі Heroku.

3.2 Огляд засобів розробки та обґрунтування їх вибору

Одним з ключових засобів розробки веб – застосунку, було обрано інтерпретовану об'єктно-орієнтовану мову програмування, Python, а саме мікрофреймовк Flask. Вибір зупинився на ньому внаслідок простої стуктури, можливості використання великої кількості ззовнішніх розширень, популярності в спільноті python – розробників, а також великої кількості навчальних матеріалів та літератури.

Для візуалізації даних, вибір зупинився на графічній бібліотеці Folium, за рахунок, її широкого спектру інструментів, сучасно візуалізованих графічних компонентів, велика кількість літератури та статтей з прикладами реалізації графічних інтерфейсів.

Для зберігання та відображення даних було обрано ,об'єктно-реляційної системи керування базою даних, PostgreSQL, внаслідок хорошої інтеграції з Python, зрозумілого синтаксису створення запитів та можливості розгорнути на хмарній платформі Heroku.

В якості, середовища розробки вибір зупинився на інтегрованому середовищі розробки для мови програмування Python, PyCharm та інструмент для роботи з базами даних, DataGrip. Даний вибір зумовлений попереднім досвідом використання, даних продуктів.

3.3 Опис основних підходів до розробки

Застосунок розроблявся з використанням REST архітектури. Representational State Transfer (REST) - це архітектурний стиль веб-сервісів, що забезпечує стандарт для обміну даних між різними типами систем. Веб-сервіси - це відкриті стандартні веб-програми, які взаємодіють з іншими програмами з мотивом обміну даними, завдяки чому вони є важливою частиною архітектури сервера клієнтів у сучасних веб-та мобільних додатках. Простіше кажучи, REST - це стандарт для обміну даними через Інтернет для взаємодії між комп'ютерними системами. Веб-сервіси, що відповідають архітектурному стилю REST, називаються веб-сервісами RESTful, які дозволяють запитувати системи доступу та маніпулювання даними, використовуючи рівномірний і визначений набір операцій без стану. REST дозволяє нам розрізняти клієнта та сервера, дозволяючи нам реалізовувати клієнта та сервер самостійно.

Найважливішою особливістю REST є підхід, який передбачає що ні клієнт, ні сервер не повинні знати стан один одного, щоб мати можливість спілкуватися. Таким чином, і клієнт, і сервер можуть зрозуміти будь-яке повідомлення, отримане без перегляду попереднього повідомлення.

Веб-сервіси у простому визначенні - це послуга, що пропонується одним електронним пристроєм на інший, що дозволяє спілкуватися через всесвітню мережу. На практиці веб-сервіси надають орієнтований на ресурси, веб-інтерфейс для сервера баз даних тощо, який використовується іншим веб-клієнтом. Веб-сервіси надають платформу для різних типів систем для спілкування один з одним, використовуючи рішення для програм, щоб мати можливість спілкуватися між собою на зрозумілій їм мові

3.4 Опис даних додатку та його інтерфейсу

Основна частина реалізації застосунку розміщена в файлі app.py. Даний файл являється корневим файлом застосунку і містить наступний функціонал:

Даний фрагмент коду відповідає за інціалізацію самої програми, запускає веб – сервер, а також вказує порт, на котрому відображається localhost:

```
if __name__ == "__main__":
    with app.test_request_context("/"):
        session["key"] = "value"
    app.run(port=5005, debug=True)
```

Також в даному файлі можна розглянути реалізацію класу RegsterForm, котрий відображає форму реєстарції, а також ініціалізує та виконує валідацію даної форми.

```
class RegisterForm(Form):
    name = StringField('Name', [validators.Length(min=5, max=50)])
    username = StringField('UserName', [validators.Length(min=3, max=30)])
    email = StringField('Email', [validators.Length(min=16, max=50)])
    password = PasswordField('Password', [
        validators.DataRequired(),
        validators.EqualTo('confirm', message='Password do not match')
    ])
    confirm = PasswordField('Confirm Password')
```

За реєстацію відповідає ендпоінт register, котрий при отриманні запиту типу пост, ініціалізує поля форми та здійснює шифрування поля password, використовуючи зовнішню бібліотеку passlib:

```
password = sha256_crypt.encrypt(str(form.password.data))
```

Наступним кроком є здійснення підключення до бази даних, яке реалізує ззовнішня бібілотека psycopg2, після чого ініціалізується об’єкт даної бібліотеки, котрий перевіряє наявність даного курсувача в базі, при негативному результаті він вносить нового користувача та завершує з’єдання, після чого користувач автоматично переноситься на строрінку авторизації

Функціонал авторизації доволі схожий своєю реалізацією, проте ввипадку успішної авторизації відбувається генерація ключа сесії.

```
if sha256_crypt.verify(password_candidate, password):
    app.logger.info('PASSWORD MATCHES')
    # passed
    session['logged_in'] = True
    session['username'] = username

    flash('You are now logged in', 'success')
```

Після чого, користувач потапряє на свою домашню сторінку.

Функція is_logged_in перевіряє чи користувач ще перебуває на сайті, відповідаючи за перевірку ключа сесії, у випадку негативного результату користувач автоматично перенаправляється на сторінку авторизації

```
def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            flash('Unauthorized, Please login', 'danger')
            return redirect(url_for('login'))

    return wrap
```

Вихід із системи здійснюється дуже просто, видаленням ключа сесії

```
@app.route('/logout')
def logout():
    session.clear()
    flash('You are now logged out', 'success')
    return redirect(url_for('login'))
```

Роут add_article реалізує отримання та вставку, даних для візуалізації, ім’я автора, та опису для майбутньої карти

```
def add_article():
    form = ArticleForm(request.form)
    if request.method == 'POST' and form.validate():
        title = form.title.data
        body = form.body.data

        # Create cursor
        try:
            connection = psycopg2.connect(user="postgres",
                                          password="12345",
                                          host="localhost",
                                          port="5432",
                                          database="postgres")
            cursor = connection.cursor()
            sqlk = """INSERT INTO postgres.test_schema.articles(title,body,author)
                           VALUES(%s,%s,%s) ;"""
            record_to_insert = (title, body, session['username'])
            cursor.execute(sqlk, record_to_insert)
            connection.commit()
```

Файл messages.html візуалізовує повідомлень системи, предназначених для інформування користувача

```
{% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    {% for category, message in messages %}
      <div class="alert alert-{{ category }}">{{ message }}</div>
    {% endfor %}
  {% endif %}
{% endwith %}

{%if error%}
  <div class="after after-danger"> {{error}}</div>
{%endif%}

{%if msg%}
  <div class="after after-success"> {{msg}}</div>
{%endif%}
```

Файл formhelper.html в свою чергу відповідає за відображення даних повідомлень.

```
{% macro render_field(field) %}
    {{ field.label }}
    {{ field(**kwargs)|safe }}
    {% if field.errors %}
        {% for error in field.errors %}
            <span class="help-inline">{{ error }} </span>
        {% endfor %}
    {% endif %}
{% endmacro %}
```

Клас _FoliumWrapper відповідає за взаємодію фреймворка Flask та графічної бібліотеки Folium. Функціонал реалізований в даному класі відповідає за ініціалізацію об’єкта карти, задання йому початкових параметрів, генерації основного шару карти в html коді , та перетворення даного об’єкта в компонент HTML-элемента, Iframe, котрий рендериться на стороні клієнта

```
class _FoliumWrapper(abc.ABC):
    """A map element that can be drawn."""
    _width = 0
    _height = 0
    _folium_map = None

    def draw(self):
        """Draw and cache map."""
        if not self._folium_map:
            self._set_folium_map()
        return self._folium_map

    def as_html(self):
        """Generate HTML to display map."""
        if not self._folium_map:
            self.draw()
        return self._inline_map(self._folium_map, self._width, self._height)

    def show(self):
        """Publish HTML."""
        IPython.display.display(IPython.display.HTML(self.as_html()))

    def _repr_html_(self):
        return self.as_html()

    @staticmethod
    def _inline_map(m, width, height):
        """Returns an embedded iframe of a folium.map."""
        html = m._repr_html_()
        return html

    @abc.abstractmethod
    def _set_folium_map(self):
```

Функціонал _Map_builder полягає в отриманні данних, за рахунок парсингу API, після чого за допомогою бібліотеки Pandas відбувається фільтрація даних, від можливих помилкових даних, після чого відфільтровані дані, візуалізовуються за дпомогою бібліотеки Folium

```
root = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19'
path_covid_data_daily = root + '/master/csse_covid_19_data/csse_covid_19_daily_reports'
data = pd.read_csv(path_covid_data_daily + '/04-11-2020.csv')
lat = data.dropna(subset=['Lat'])
lon = data.dropna(subset=['Long_'])
confirmed = data['Confirmed']
```

```
for i in range(0, len(data)):
    if pd.np.isnan(data.iloc[i]['Long_']) or pd.np.isnan(data.iloc[i]['Lat']) or confirmed.iloc[i] == 0:
        continue
```

```
map = folium.Map(location=[37.296933, -121.9574983], zoom_start=5, tiles="CartoDB dark_matter")
folium.CircleMarker(
    location=[data.iloc[i]['Lat'], data.iloc[i]['Long_']],
    radius=2, popup=str(confirmed.iloc[i]) + " m",
    fill_color=color_change(confirmed.iloc[i]), color="gray", fill_opacity=0.9).add_to(
    marker_cluster)
```

Висновок

В процесі дослідження теми було прийнто рішення розробити веб - застосунок для візуаліції даних В результаті виконаної роботи мета була досягнута, а постановка задачі дотримана. Розроблений застосунок відображає принципи розробки веб-додатків, з викритсанням мікрофреймворку Flask та демонструє гнучкість та актуальність графічної бібліотеки Folium

Посилання на використані джерела

https://en.wikipedia.org/wiki/World_Wide_Web

https://habr.com/ru/post/215117/

https://habr.com/ru/post/50147/

https://www.lifewire.com/servers-in-computer-networking-817380

https://www.sitepoint.com/domain-name-www-or-not/

https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed

https://www.sas.com/en_us/insights/big-data/data-visualization.html#history

https://www.klipfolio.com/resources/articles/what-is-data-visualization

https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/10471/461.pdf?sequence=3 History of python

https://towardsdatascience.com/complete-guide-to-data-visualization-with-python-2dd74df12b5e

https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed

https://towardsdatascience.com/data-visualization-with-bokeh-in-python-part-one-getting-started-a11655a467d4

https://towardsdatascience.com/data-101s-spatial-visualizations-and-analysis-in-python-with-folium-39730da2adf

file:///C:/Users/DS/Downloads/Mastering%20Social%20Media%20Mining%20with%20Python.pdf