МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра Системного проектування

Допустити до захисту

Завідувач кафедри

_____ доц.Шувар Роман Ярославович

(підпис) (ПІБ)

Кваліфікаційна робота

Бакалавр

(освітній ступінь)

на тему **″ Розробка архітектури для поєднання мови програмування Python з технологією Javascript CS5 ″**

Виконав:

студент групи ФеП-41

спеціальності:

121 Інженерія програмного забезпечення

_____ Безсонов Кирило Володимирович

(підпис) (ПІБ)

Науковий керівник:

_____ ас.Мисюк Ірина Володимірівна

(підпис) (ПІБ)

Рецензент:

_____Болеста І.М.

(підпис) (ПІБ)

Львів 2023

Анотація

Безсонов К.В. Розробка архітектури для поєднання мови програмування Python з технологією Javascript CS5. Дипломна робота за спеціальністю 121 Інженерія програмного забезпечення – ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА, Львів, 2023 рік.

Наш проект має на меті дослідити можливості мови програмування Python у створенні веб-додатків.Планується досліджувати різні інструменти та фреймворки, які дозволяють створювати веб-додатки з використанням Python. Для цього буде проведено аналіз існуючих бібліотек та фреймворків, що дозволяють розробляти веб-додатки на Python, та проаналізуємо їх переваги та недоліки.

Ключові слова: Python, Flask, Back-end, сервер, Bootstrap, бази данних.

Abastract

Bezsonov Kyrylo. Developing an architecture for combining the Python programming language with Javascript CS5 technology. Diploma thesis in the speciality 121 Software Engineering - Ivan Franko National University of Lviv, Lviv, 2023.

Our project aims to explore the possibilities of the Python programming language in creating web applications. We plan to explore various tools and frameworks that allow you to create web applications using Python. To do this, we will analyse the existing libraries and frameworks that allow developing web applications in Python and analyse their advantages and disadvantages.

Keywords: Python, Flask, Back-end, server, Bootstrap, databases.

Зміст

Перелік умовних позначень 5

Вступ 6

Розділ 1. Теоретична частина. Актуальність теми 7

1.1. Актуальність веб-розробки на Python та Javascript 7

1.2. Тенденції веб-розробки на Python 7

Розділ 2. Опис та перелік використаних технологій 10

2. Середовище програмування та контролю версій, програми для керування базами даних 10

2.1. Visual Studio Code 10

2.2. Переваги розробки на Python в Visual Studio Code 10

2.3. Система контролю версій Git 11

2.4. Веб-сервіс GitHub 11

2.5. Інтерфейс для роботи з базами даних DB Browser (SQLite) 12

2.6. Компонент WSL 12

3. Технології для створення проекту 14

3.2. Фреймворк Flask 14

3.3. Розширення фреймворку Flask-login 16

3.4. Розширення фреймворку Flask-migrate 16

3.5. Розширення фреймворку Flask-SQLAlchemy 17

3.6. Розширення фреймворку Flask-WTF 17

3.7. Бібліотека SQLAlchemy 18

3.8. Відкрите програмне забезпечення Redis 19

3.9. Система Celery 19

3.10. Бібліотека Alembic 20

3.11. Бібліотека BeautifulSoup4 21

3.12. Бібліотека Werkzeug 21

3.13. Мова розмітка HTML 22

3.14. Мова стилів CSS 22

3.15. Мова JavaScript 23

3.16. Bootstrap 24

Розділ 3. Розробка вебсайту. Створення функціоналу застосунку. 26

4. Розпочнемо з створення простого веб-застосунку на Flask 26

4.1. Імплементуємо API для відображення погоди у місті Львів 26

4.2. Застосуємо фреймворк Bootstrap 28

4.3. Реалізуємо функцію збору новин з стороннього сайту 28

4.4. Реалізація збору контенту з отриманих раніше публікацій 29
4.5. Робота з базою даних 31
4.6. Створення моделей в базі даних 32
4.7. Реалізація авторизації та перевірки прав користувача 33
4.8. Функціонал запам’ятовування користувача 38
4.9. Розбиття проекту на модулі та використання Blueprint 38
4.10. Скрипт для запуску додатку 40
4.11. __init__.py 41
4.12. Автоматизація збору новин, Redis&Celery 42
4.13. Реалізація функціоналу коментарів 44
Висновки: 47
Використані джерела 48

Перелік умовних позначень

API - *Application Programming Interface*, Прикладни́й програ́мний інтерфе́йс

Render - *візуалізація, проявлення, відмальовування, подання*

Task – Завдання, задача

CDN – Content Delivery Network, Мережа доправлення контенту

URL – *Uniform Resource Locator, єдиний вказівник на ресурс*

Парсинг - це процес автоматичного збору даних та їх структурування

Вступ

У даній дипломній роботі досліджується використання мов Python та Javascript у веб-розробці. Дана дипломна робота є актуальною та важливою в контексті сучасних тенденцій розвитку веб-технологій.

В результаті проведеного дослідження та розробки проекту, очікується покращення розуміння можливостей та особливостей використання мов Python та Javascript у веб-розробці.

Основна увага зосереджена на вивченні фреймворків мови Python та їх застосуванні у розробці проекту.

Метою роботи є набуття нових навичок та здатності їх використовувати на практиці.

Для досягнення мети було поставлено ряд завдань:

1. Ознайомлення з документацією фреймворків мови Python та Javascript для веб-розробки
2. Дослідження можливостей, переваг та недоліків
3. Здобуття практичних навичок
4. Створення веб-проекту

Проект було завантажено на GitHub – https://github.com/killreal2002/diplom

Розділ 1. Теоретична частина. Актуальність теми

1.1. Актуальність веб-розробки на Python та Javascript

Тема актуальності даної дипломної роботи базується на швидкому розвитку веб-технологій та необхідності використання мов програмування, що дозволяють створювати високоякісні та масштабовані веб-додатки. Python та Javascript є одними з найбільш популярних мов програмування для веб-розробки в наш час, і вони постійно розвиваються та оновлюються.Основна мета даної роботи полягає в тому, щоб дослідити та вивчити нові технології, що забезпечують розробку веб-додатків з використанням мов програмування Python та Javascript, та застосувати ці знання на практиці.

Завдяки цьому можливо досягти більш високої продуктивності та ефективності веб-розробки, що є надзвичайно важливим у конкурентному середовищі сучасного ринку технологій.

Також, зростає значення веб-розробки для підтримки бізнес-процесів різноманітних компаній та організацій. За допомогою веб-додатків можна забезпечити зручний та швидкий доступ до інформації та послуг для клієнтів та співробітників.

Таким чином, дослідження та застосування новітніх технологій веб-розробки може сприяти покращенню ефективності бізнес-процесів та задоволенню потреб користувачів. За останні роки веб-розробка стала все більш складною та різноманітною галуззю, що вимагає від розробників постійного вдосконалення та вивчення нових інструментів. Мови програмування Python та Javascript є дуже популярними в галузі веб-розробки, оскільки вони дозволяють швидко та ефективно розробляти складні веб-додатки та сервіси.

Крім того, Python та Javascript мають велику кількість фреймворків, які дозволяють значно спростити розробку веб-додатків та забезпечити їх ефективну роботу. Навички роботи з цими мовами та їх фреймворками стали дуже цінними для веб-розробників, оскільки їх допомагають створювати складні веб-додатки, що відповідають сучасним вимогам.

Отже, дослідження та вивчення мов програмування Python та Javascript є дуже актуальними в сучасних умовах та можуть допомогти розробникам стати більш ефективними та конкурентноздатними на ринку веб-розробки.

1.2. Тенденції веб-розробки на Python

Веб-розробка на мові Python є однією з найбільш популярних областей використання цієї мови програмування. Останнім часом спостерігається тенденція до розвитку веб-фреймворків на базі Python, таких як Django, Flask, Pyramid і багатьох інших.

Однією з особливостей веб-розробки на Python є широкий вибір бібліотек та інструментів, що дозволяє розробникам швидко та ефективно створювати веб-додатки з різноманітними функціональними можливостями. Крім того, Python підтримує багатопоточність та асинхронний код, що робить його особливо привабливим для веб-розробки в умовах високих навантажень.

Використання Python у веб-розробці також дозволяє розробникам писати код швидше та з меншою кількістю помилок завдяки його зрозумілому синтаксису та високій якості документації. Крім того, Python є відкритою мовою з великою спільнотою розробників, що сприяє швидкому розвитку та підтримці веб-фреймворків.

Отже, веб-розробка на Python є перспективною галуззю, що забезпечує широкі можливості та ефективність в розробці веб-додатків різного рівня складності.

У нашу цифрову епоху веб-розробка стала вирішальним аспектом для будь-якого бізнесу чи організації, які потребують мати присутність в Інтернеті. Python, Flask та Bootstrap - популярні інструменти, що використовуються для веб-розробки,

які мають важливе значення в цій галузі. Простота та універсальність Python роблять його улюбленою мовою для розробників. Flask - це легкий та гнучкий веб-фреймворк, який дозволяє розробникам легко створювати веб-додатки. Bootstrap, з іншого боку, є потужним фронтенд-фреймворком, який забезпечує крос-платформну сумісність. Разом ці інструменти є незамінними для веб-розробки, і їх значення, як очікується, зростатиме в майбутньому.

Оскільки попит на веб-додатки продовжує зростати додатки продовжує зростати, зростає потреба в розробниках, які володіють цими платформами, оскільки вони пропонують ефективні рішення для створення складних і динамічних веб-сайтів вони пропонують ефективні та спрощені рішення для створення складних і динамічних веб-сайтів. Тому для розробників-початківців дуже важливо для розробників-початківців дуже важливо визнати цінність цих інструментів та інвестувати в їх вивчення і використання їх для задоволення мінливих потреб цифрової епохи.

Веб-розробка на Python за останні роки набула величезної популярності завдяки своїй простоті, гнучкості та сумісності з численними веб-фреймворками, такими як Flask та Bootstrap. Зі зростанням попиту на веб-додатки та потребою в ефективних і масштабованих фреймворках для веб-розробки фреймворків, Python став широко розповсюдженою мовою програмування у сфері веб-розробки. Таким чином, актуальність і важливість Python, Flask і Bootstrap у веб-розробці неможливо переоцінити, і очікується, що вони будуть продовжувати процвітати в майбутньому.

Веб-розробка на Python стає все більш популярною, оскільки мова має багато переваг, таких як велика кількість бібліотек та фреймворків, висока продуктивність, широка підтримка спільноти та зручність у розробці.

Однак, також важливо пам'ятати про особливості використання мови Python в веб-розробці, такі як необхідність налагодження конфігурації сервера та підтримка веб-серверів.

Крім того, не слід забувати про важливість знання інших технологій, таких як Javascript, яка залишається основною мовою для клієнтської веб-розробки.

У цілому, розуміння можливостей та особливостей використання різних мов та технологій у веб-розробці може допомогти розробникам створювати більш ефективні та функціональні веб-додатки.

Розділ 2. Опис та перелік використаних технологій

2. Середовище програмування та контролю версій, програми для керування базами даних

2.1. Visual Studio Code

Visual Studio Code (VS Code) - це безкоштовний, відкритий і легкий редактор коду, розроблений компанією Microsoft. Він підтримує багато мов програмування, включаючи JavaScript, TypeScript, Python, C++, Java, Go, Ruby, і багато інших.

VS Code має багато корисних функцій, включаючи автодоповнення коду, наступення за посиланнями (code navigation), відладка коду, підтримку Git і роботу з розширеннями [4].

Одна з найбільших переваг VS Code - це можливість розширення його функціональності за допомогою додаткових плагінів. Наявність великої кількості розширень дозволяє налаштувати редактор на свій смак, забезпечуючи підтримку різноманітних технологій, фреймворків та інших інструментів.

VS Code є переносним на різні операційні системи, такі як Windows, MacOS і Linux. Редактор працює досить швидко, що робить його популярним серед програмістів.

Крім того, VS Code інтегрується з багатьма іншими інструментами для розробки, такими як системи контролю версій (наприклад, Git), засоби автоматизації завдань (наприклад, Gulp або Grunt), сервіси хмарного зберігання та багато інших.

У загальному, VS Code - це потужний інструмент для розробки програмного забезпечення з великою кількістю корисних функцій і підтримкою великої кількості мов програмування, що дозволяє розробникам створювати якісний код швидко і ефективно.

2.2. Переваги розробки на Python в Visual Studio Code

Розширення Python: VS Code має розширення Python, яке дозволяє створювати, редагувати та відлагоджувати код Python, а також запускати його з середовища VS Code. Це дозволяє зробити VS Code ідеальним вибором для розробників Python.

Підтримка відлагодження: VS Code має вбудовану підтримку відлагодження для Python, що дозволяє легко відстежувати та виправляти помилки у коді. Це забезпечує більш ефективний процес розробки.

Інтерактивна консоль: VS Code дозволяє запускати Python код у вбудованій інтерактивній консолі, що забезпечує більш швидкий та простий спосіб експериментувати з кодом та розв'язувати проблеми.

Підтримка Git: VS Code має вбудовану підтримку Git, що дозволяє легко зберігати та керувати версіями коду Python. Це забезпечує більш ефективний процес роботи з кодом та сприяє більшій продуктивності.

Розширення та налаштування: VS Code має широкий вибір розширень та можливостей налаштування, що дозволяє розширювати функціональність та адаптувати середовище до потреб розробника.

Узагалі, VS Code є потужним та гнучким інструментом для розробки на Python, що забезпечує більш ефективний та продуктивний процес розробки.

2.3. Система контролю версій Git

Git - це розподілена система керування версіями, що дозволяє відстежувати зміни в програмному коді та спільно працювати над проектами з іншими розробниками. Одним з основних переваг Git є його здатність ефективно керувати гілками (branches) в репозиторії, що дозволяє розробляти різні функціональності у окремих гілках та легко злити їх з головною гілкою (master) після тестування [6].

Кожний розробник може мати свій локальний репозиторій з Git, на якому він може працювати над своїми змінами в коді, додавати нові файли та комітити їх (створювати знімки змін). Після цього він може відправити свої зміни віддаленому репозиторію (remote repository), що може знаходитись на сервісах, таких як GitHub, GitLab або Bitbucket.

Git дозволяє вирішувати конфлікти в змінах, що вносяться різними розробниками, та забезпечує можливість повернення до попередніх версій коду в разі помилок чи проблем з новими змінами. Крім того, Git дозволяє вести статистику по змінам у коді та забезпечує можливість перегляду історії змін, коментарів до змін та авторів змін.

Git є однією з найбільш популярних систем керування версіями серед розробників і використовується для керування версіями коду в різних проектах та комерційних продуктах.

2.4. Веб-сервіс GitHub

GitHub - це веб-сервіс, який надає безкоштовне зберігання репозиторіїв Git [7]. GitHub дозволяє розробникам зберігати та спільно працювати над проектами з використанням Git, надаючи зручний інтерфейс для перегляду та зміни коду, керування змінами, відстеження проблем, побудови проектів та іншого.

У GitHub можна створювати репозиторії як для відкритих, так і для приватних проектів, надаючи зручний механізм керування доступом для співробітників. Також

GitHub має інтеграцію з багатьма іншими інструментами розробки, такими як Travis CI, Code Climate, ZenHub, і багатьма іншими.
Крім того, GitHub дозволяє розробникам зберігати документацію, розробляти власні проекти та знаходити проекти інших розробників, що може сприяти розвитку різноманітних проектів та відкритому співробітництву.

2.5. Інтерфейс для роботи з базами даних DB Browser (SQLite)[5]

DBBrowser - це безкоштовна і відкрита програма для керування базами даних на комп'ютері. З її допомогою можна підключатися до різних типів баз даних, створювати, видаляти та редагувати таблиці, запити та інші об'єкти баз даних.

Інтерфейс програми простий та зручний у використанні, тому її можуть використовувати як досвідчені, так і новачки в галузі баз даних. Крім того, DBBrowser підтримує відображення даних у вигляді таблиць, діаграм, графіків та інших візуальних елементів, що полегшує аналіз та обробку інформації.

DBBrowser підтримує роботу з такими базами даних, як SQLite, MySQL, PostgreSQL, Oracle та інші. Програма має вбудований SQL-редактор, що дозволяє писати запити на мові SQL та виконувати їх безпосередньо в програмі. Крім того, DBBrowser підтримує експорт та імпорт даних у різних форматах, що дозволяє переміщувати дані між різними базами даних та програмами.

Загалом, DBBrowser є потужним та зручним інструментом для керування базами даних на комп'ютері, який може бути корисним для розробників, аналітиків даних та інших фахівців, які працюють з базами даних.

2.6. Компонент WSL

WSL (Windows Subsystem for Linux) - це компонент, що доступний в операційній системі Windows 10 і пізніших версіях, який дозволяє виконувати середовище Linux безпосередньо на Windows [12]. Використовуючи WSL, користувачі можуть запускати Linux-програми, виконувати команди та отримувати доступ до файлів Linux-системи прямо зі свого комп'ютера з Windows.

WSL забезпечує високий рівень сумісності між Linux і Windows шляхом використання ядра Linux, що запускається в межах віртуальної машини. Це дозволяє запускати багато популярних дистрибутивів Linux, таких як Ubuntu, Debian, Fedora і багато інших, на комп'ютері з Windows без необхідності встановлення окремого віртуального середовища.

WSL надає ряд переваг користувачам, які хочуть використовувати Linux-інструменти або розробляти для платформи Linux, але залишатися в середовищі Windows. Основні переваги WSL включають:

Легке встановлення: WSL можна легко встановити з Microsoft Store або з використанням командної строки.

Висока сумісність: WSL надає високий рівень сумісності зі стандартами Linux, дозволяючи запускати багато програм і інструментів Linux без проблем.

Доступ до файлів: Користувачі можуть отримати доступ до файлів Linux-системи прямо з провідника Windows або з командного рядка, що дозволяє зручно працювати зі спільними файлами.

Інтеграція зі середовищем Windows: WSL інтегрується зі середовищем Windows, дозволяючи використовувати Linux-інструменти в поєднанні з Windows-програмами та сервісами.

Розширені можливості розробки: З використанням WSL, розробники можуть легко використовувати популярні інструменти розробки Linux, такі як командний рядок, текстові редактори, компілятори і інші, прямо в середовищі Windows.

WSL став популярним серед розробників і користувачів, які хочуть поєднати переваги обох світів - Linux і Windows. Цей компонент дозволяє більш гнучко працювати з різними платформами і інструментами, відкриваючи нові можливості для розробників та користувачів.

3. Технології для створення проекту

3.1. Мова Python

Python - це інтерпретована мова програмування високого рівня зі строгою динамічною типізацією та об'єктно-орієнтованою парадигмою, розроблена Гвідо ван Россумом у 1990 році. Багато високорівневих структур даних та динамічна семантика забезпечують швидку розробку програм, а пакети модулів дозволяють модульність та повторне використання коду. Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану. Python доступний на всіх основних платформах у вигляді інтерпретатора та стандартної бібліотеки у скомпільованій та вихідній формі [8].

Деякі переваги Python включають:

Чистий синтаксис, що дозволяє виділяти блоки за допомогою відступів.

Переносність програм, що характерно для більшості інтерпретованих мов.

Велика кількість корисних модулів у стандартному дистрибутиві, включаючи модуль для розробки графічного інтерфейсу.

Можливість використання Python у діалоговому режимі для експериментування та розв'язання простих задач.

Просте та потужне середовище розробки IDLE, написане мовою Python.

Зручні засоби для розв'язання математичних проблем, включаючи роботу з комплексними числами та цілими числами довільної величини.

Відкритий код, що дозволяє його редагувати іншими користувачами.

Python також має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Завдяки елегантному синтаксису, динамічній обробці типів та інтерпретації, Python є ідеальним для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python та багата Стандартна бібліотека можуть бути отримані з сайту Python www.python.org і розповсюджуватися вільно. Крім того, Python може бути розширений функціями та типами даних, розробленими на C чи C++, а також використаний як мова розширення для прикладних програм, що потребують подальшого налагодження.

3.2. Фреймворк Flask

Flask - це легкий фреймворк для розробки веб-додатків на мові програмування Python. Flask надає базові інструменти для створення веб-додатків, такі як маршрутизація URL-адрес, обробка запитів, шаблонізація і підтримка RESTful API [9].

Flask простий у використанні та дозволяє розробникам швидко створювати веб-додатки з мінімальними витратами. Він не нав'язує певної архітектури або структури проекту, тому розробник може самостійно вибрати найбільш підходящий для нього підхід.

Flask також має велику кількість розширень (extensions) для роботи з базами даних, формами, аутентифікацією, розгортанням та іншого, що дозволяє розширити функціональність додатку без необхідності вручну розробляти весь необхідний код.

Загалом, Flask є дуже популярним фреймворком для розробки веб-додатків на Python, завдяки своїй простоті та гнучкості.

Основні переваги фреймворку Flask для розробки веб-додатків на Python:

- Легкість в освоєнні і використанні. Flask має просту структуру, зручний API і добре документований код, що дозволяє швидко освоювати розробку веб-додатків на Python.
- Гнучкість і розширюваність. Flask пропонує широкий вибір доповнень та розширень, що дозволяє легко налаштувати фреймворк під конкретні потреби проекту.
- Мінімальна кількість залежностей. Flask має мінімальну кількість залежностей, що забезпечує легкість встановлення і розгортання веб-додатків на сервері.
- Швидкість розробки. Flask дозволяє швидко розробляти прототипи веб-додатків і відлагоджувати їх, завдяки простоті і легкості використання.
- Підтримка різних технологій. Flask має вбудовану підтримку різних технологій, таких як Jinja2, для роботи з шаблонами сторінок, та Werkzeug, для обробки HTTP-запитів і відповідей.
- Хороша спільнота користувачів. Flask має широку спільноту користувачів, що забезпечує швидку підтримку і вирішення проблем, а також розробку нових доповнень і розширень.

Основні недоліки Flask наступні:

- Не входить до складу стандартної бібліотеки Python: Flask не є частиною стандартної бібліотеки Python, що означає, що для використання Flask потрібно окремо встановлювати пакет.
- Обмежена функціональність: Flask має обмежену функціональність порівняно з більш складними фреймворками, такими як Django, що може бути недостатньо для великих та складних проектів.
- Невелика кількість вбудованих інструментів безпеки: Flask має лише декілька вбудованих інструментів безпеки, що може призвести до потенційних проблем з безпекою, якщо розробник не буде уважним.
- Відсутність стандартів: Flask не має жорстких стандартів та настанов щодо організації структури проекту, що може призвести до складнощів з розробкою та підтримкою проектів.
- Відповідальність за вибір додаткових компонентів: Flask не має вбудованих функцій для багатьох задач, тому розробник повинен відповідально підходити до вибору додаткових компонентів та бібліотек для реалізації певних функцій. Це може збільшити складність проекту та зробити його вразливим до помилок.

3.3. Розширення фреймворку Flask-login

Flask-Login - це розширення фреймворку Flask для автентифікації користувачів. Це дозволяє легко виконувати різні завдання, пов'язані з управлінням авторизацією та сеансами користувачів у вашому веб-додатку [20].

Зокрема, Flask-Login надає інструменти для зберігання інформації про користувачів у сеансах, перевірки даних для авторизації користувача, захисту сторінок від неавторизованого доступу, а також для роботи з різними типами авторизації (наприклад, пароль, електронна пошта або соціальні мережі).

Завдяки Flask-Login, розробники можуть зосередитися на функціональності своїх додатків, не заморочуючись з реалізацією авторизації та зберіганням даних користувачів. Розширення забезпечує гнучкі налаштування та легку інтеграцію з іншими компонентами Flask.

Якщо вам потрібна авторизація користувачів в вашому Flask-додатку, Flask-Login може бути чудовим вибором для забезпечення безпеки та зручності.

3.4. Розширення фреймворку Flask-migrate

Flask-Migrate - це розширення для Flask, яке дозволяє легко керувати міграціями баз даних, які використовують SQLAlchemy. Міграції баз даних - це процес збереження структурної цілісності бази даних під час розширення або зміни моделей даних [22].

З Flask-Migrate ви можете легко створювати нові міграції та застосовувати їх до бази даних. Він забезпечує відстеження змін моделей даних та автоматичне створення SQL-запитів для збереження змін в базі даних. Крім того, Flask-Migrate дозволяє перевіряти статус міграцій та робити зворотні міграції.

Flask-Migrate використовує Alembic, потужну бібліотеку міграції баз даних для Python. Alembic дозволяє зберігати версії моделей даних та автоматично створювати SQL-запити для змін в базі даних. Flask-Migrate дозволяє просто використовувати Alembic для Flask-додатків.

Загалом, Flask-Migrate дозволяє зберігати структурну цілісність бази даних та дозволяє легко створювати та застосовувати міграції, що робить його корисним розширенням для будь-якого Flask-додатку, який використовує SQLAlchemy.

3.5. Розширення фреймворку Flask-SQLAlchemy

Flask-SQLAlchemy - це розширення Flask, яке надає інструменти для взаємодії з базою даних. Воно підтримує різні бази даних, включаючи SQLite, MySQL та PostgreSQL [21].

Розширення дозволяє легко встановлювати зв'язок з базою даних і створювати моделі, які можуть використовуватися для збереження даних. Моделі є класами, які відображають таблиці бази даних, і вони містять поля, які відповідають стовпцям таблиці.

Flask-SQLAlchemy надає ряд корисних інструментів для роботи з базою даних, включаючи підтримку транзакцій, міграцій, пошуку, фільтрації та сортування. Воно також дозволяє використовувати різні типи полів для моделей, включаючи текстові поля, числові поля, булеві поля, дати та часи.

Flask-SQLAlchemy є потужним інструментом для розробки веб-додатків, що використовують бази даних. Воно дозволяє легко взаємодіяти з базою даних та створювати складні запити для отримання та збереження даних.

3.6. Розширення фреймворку Flask-WTF

Flask-WTF є розширенням Flask для роботи з формами. Flask-WTF дозволяє легко створювати та валідувати HTML-форми в Flask додатках. Воно базується на WTForms, що є бібліотекою Python для роботи з формами [19].

З Flask-WTF, ви можете легко створювати HTML-форми в Python-коді та використовувати їх у вашому Flask додатку. Flask-WTF надає ряд функцій для валідації введення користувача та захисту від CSRF атак.

Основні функції Flask-WTF включають:

Підтримка різних типів полів форми, таких як текстові поля, чекбокси, радіокнопки, випадаючі списки тощо;

Валідація форм, яка дозволяє перевіряти правильність введення даних користувачем перед їх збереженням;

Підтримка роботи з файлами;

Захист від CSRF атак за допомогою включення в форму спеціального поля з токеном.

Flask-WTF є потужним інструментом для роботи з формами в Flask додатках та дозволяє значно спростити розробку та валідацію HTML-форм.

3.7. Бібліотека SQLAlchemy

SQLAlchemy - це бібліотека мови Python, призначена для роботи з реляційними базами даних [3]. Вона дозволяє програмістам працювати з базами даних на вищому рівні абстракції, що дозволяє знизити складність написання SQL-запитів та підтримки коду, що взаємодіє з базою даних.

SQLAlchemy складається з двох основних компонентів: ORM (Object-Relational Mapping) та Core. ORM є вищим рівнем абстракції та дозволяє програмістам працювати з базами даних, як з об'єктами Python, тобто моделювати структуру таблиць в класах Python. Core, натомість, пропонує більш низькорівневий інтерфейс для роботи з базами даних, але дозволяє використовувати більш складні запити та підтримує більш широкий спектр баз даних.

SQLAlchemy підтримує різні типи баз даних, включаючи PostgreSQL, MySQL, Oracle, SQLite та багато інших. Крім того, вона дозволяє програмістам використовувати різні рівні ізоляції транзакцій, кешування запитів та підтримує безпеку на рівні бази даних, таку як шифрування даних.

Загалом, SQLAlchemy є потужним інструментом для роботи з базами даних в мові Python та дозволяє зменшити зусилля, необхідні для написання коду, що взаємодіє з базою даних, та зберігає код більш зрозумілим та легким для підтримки.

3.8. Відкрите програмне забезпечення Redis

Redis (Remote Dictionary Server) - це відкрите програмне забезпечення, яке використовується як віддалена пам'ять з високою продуктивністю та надійністю [16]. Він працює як ключ-значення (key-value) база даних, де дані зберігаються в пам'яті для швидкого доступу.

Основні особливості Redis:

Швидкість: Redis є дуже швидким інструментом завдяки тому, що дані зберігаються в оперативній пам'яті, що дозволяє виконувати дуже швидкі операції зчитування та запису. Він може обробляти мільйони операцій на секунду.

Підтримка різних типів даних: Redis підтримує різні типи даних, включаючи рядки, хеші, списки, множини, сортовані множини та бітові рядки. Це дає можливість зберігати і маніпулювати різнорідними даними в одному місці.

Наявність: Redis підтримує механізми реплікації та кластеризації, що забезпечує високу наявність та надійність. Він може автоматично реплікувати дані на декількох серверах, забезпечуючи резервні копії та здатність до відновлення.

Повний контроль: Redis надає розширені можливості керування даними, включаючи транзакції, операції блокування та інші механізми, що дозволяють забезпечити консистентність та безпеку даних.

Підтримка розширень: Redis має багато розширень, які додають додаткові функціональні можливості, такі як геопросторові запити, повідомлення черги, індекси для повнотекстового пошуку та багато інших.

Redis використовується в багатьох сферах, зокрема веб-розробці, аналітиці даних, кешуванні, сесіях користувачів, реал-тайм аналітиці, чергах повідомлень та багатьох інших сценаріях, де важлива швидкодія та надійність доступу до даних.

3.9. Система Celery

Celery - це розподілена система обробки завдань (task queue), написана на мові програмування Python. Вона призначена для асинхронного виконання завдань у фоновому режимі та розподіленого обчислення [2].

Основні особливості Celery:

Асинхронні завдання: Celery дозволяє відокремити обробку довгих або ресурсоємних завдань від основного веб-додатку. Ви можете відправляти завдання в чергу, і Celery буде обробляти їх асинхронно, звільняючи основний потік веб-застосунку для обробки нових запитів.

Розподілена система: Celery підтримує розподілену обробку завдань, що означає, що ви можете мати декілька воркерів (workers), які виконують завдання на різних машинах або в різних контейнерах. Це дозволяє горизонтально масштабувати обробку завдань, забезпечуючи швидкість та масштабованість.

Планування та періодичні завдання: Celery має вбудовану можливість планування завдань на виконання у певний час або з певною періодичністю. Ви можете налаштувати періодичні завдання, щоб вони автоматично виконувалися за заданим графіком, що дуже зручно для автоматизації повторюваних завдань.

Результати та стан завдань: Celery зберігає стан завдань та їх результати, що дозволяє вам отримувати результати виконання завдань та контролювати їх стан. Ви можете отримувати інформацію про прогрес виконання завдань, відстежувати помилки та здійснювати ретрі-повтори в разі необхідності.

Розширення та інтеграція: Celery має багатий екосистему розширень та інтеграцій з різними інструментами та фреймворками. Ви можете використовувати Celery разом з Django, Flask, Pyramid та багатьма іншими фреймворками, а також інтегрувати його з іншими технологіями, такими як Redis або RabbitMQ.

Загалом, Celery є потужним інструментом для асинхронного виконання завдань у фоновому режимі. Він надає широкі можливості для розподіленої обробки завдань та дозволяє покращити швидкодію та масштабованість вашого додатку.

3.10. Бібліотека Alembic

Alembic - це Python бібліотека для управління версіями схеми бази даних. Вона дозволяє розробникам створювати та зберігати міграції бази даних у вигляді Python коду, що дає можливість робити зміни схеми бази даних і контролювати їх управління [1].

За допомогою Alembic можна легко керувати схемою бази даних на різних етапах розробки проекту, зокрема створювати таблиці, змінювати їх поля, додавати та видаляти стовпці, індекси, зв'язки між таблицями тощо.

Однією з основних переваг Alembic є можливість автоматичного генерування міграцій на основі змін у моделях бази даних, що дозволяє значно спростити розробку та підтримку бази даних.

Крім того, Alembic підтримує керування версіями міграцій, що дозволяє точно визначити стан бази даних на будь-якому етапі розробки.

Загалом, Alembic є потужним інструментом для керування версіями схеми бази даних в проектах на Python. Він дозволяє легко робити зміни в базі даних та зберігати їх в контрольованому середовищі, що є особливо важливим для великих та складних проектів.

3.11. Бібліотека BeautifulSoup4

BeautifulSoup4 є однією з найпопулярніших бібліотек для парсингу HTML і XML даних в мові програмування Python [18]. Вона надає простий та зрозумілий спосіб витягування даних з веб-сторінок шляхом розбору їх структури.

Основна перевага BeautifulSoup4 полягає в його простоті використання. Він надає інтуїтивно зрозумілі методи та функції, які дозволяють зручно навігувати по структурі HTML-документа, знаходити та витягувати потрібні елементи, отримувати їх вміст і атрибути. Завдяки цьому, розробникам не потрібно писати складні алгоритми для обробки HTML-даних, і вони можуть швидко отримувати необхідну інформацію зі сторінок.

Крім того, BeautifulSoup4 підтримує різні парсери HTML, такі як "html.parser", "lxml" і "html5lib", що дозволяє вибрати найкращий парсер для конкретних потреб. Кожен парсер має свої переваги та недоліки з точки зору швидкодії та сумісності з різними типами HTML-документів.

Також варто відзначити, що BeautifulSoup4 добре справляється з неправильно відформатованими або недійсними HTML-даними, намагаючись інтерпретувати їх наскільки це можливо. Це робить його гнучким і придатним для роботи з різноманітними джерелами даних.

Узагалі, BeautifulSoup4 є потужним інструментом для витягування даних з HTML-сторінок в середовищі Python. Він допомагає забезпечити швидку та зручну обробку веб-даних, що робить його популярним вибором серед розробників.

3.12. Бібліотека Werkzeug

Werkzeug є однією з найпопулярніших бібліотек для роботи з веб-фреймворками в мові програмування Python. Вона надає потужні інструменти для обробки HTTP-запитів та відповідей, маршрутизації URL, керування сесіями, роботи з кукісами та багатьма іншими веб-задачами [17].

Одна з основних переваг Werkzeug полягає в його простоті використання та зрозумілому API. Він надає легкий спосіб створити власний веб-сервер або інтегруватися з існуючими веб-фреймворками, такими як Flask. Завдяки простоті та зрозумілості інтерфейсу Werkzeug дозволяє швидко розробляти веб-додатки та зосередитися на їх функціональності.

Werkzeug також забезпечує низку важливих функцій, які допомагають забезпечити безпеку веб-додатків. Він має вбудовану підтримку для обробки кросс-сайтових запитів (CSRF), підпису та перевірки кукісів, аутентифікації та авторизації користувачів, що дозволяє розробникам створювати безпечні веб-додатки з меншими зусиллями.

Крім того, Werkzeug має добре документовану API, яка спрощує процес розробки і дозволяє розробникам швидко зорієнтуватися в функціональності бібліотеки. Вона також має велику спільноту користувачів та активно розвивається, що забезпечує підтримку та оновлення.

Загалом, Werkzeug є потужним інструментом для розробки веб-додатків в мові програмування Python. Він надає простоту використання, безпеку, гнучкість та широкі можливості для роботи з веб-фреймворками, допомагаючи розробникам створювати високоякісні веб-додатки.

3.13. Мова розмітка HTML

HTML (HyperText Markup Language) є стандартною мовою розмітки для створення веб-сторінок. Вона використовується для структурування та оформлення вмісту веб-сторінок, зокрема тексту, зображень, посилань, таблиць і форм [13].

HTML базується на використанні тегів (tags), які визначають різні елементи веб-сторінки. Теги починаються з відкриваючого тегу <tag> і завершуються закриваючим тегом </tag>. Вміст елемента розміщується між цими тегами.

HTML дозволяє визначати заголовки, абзаци, списки, таблиці, зображення, посилання, форми та багато іншого. Кожен елемент має свою семантику та може мати додаткові атрибути для встановлення різних властивостей.

Одна з основних функцій HTML полягає у встановленні структури веб-сторінки. Вона дозволяє організувати вміст за розділами, надає заголовки різного рівня, створює списки для переліку елементів, та використовує таблиці для організації табличних даних.

Крім того, HTML використовується для форматування вмісту. За допомогою CSS (Cascading Style Sheets) можна задавати стилі, кольори, розміри шрифтів, відступи, рамки та інші властивості для елементів HTML.

HTML є основою для розробки веб-сторінок та веб-додатків. Вона використовується разом з CSS і JavaScript для створення багатофункціональних, інтерактивних та естетично привабливих веб-інтерфейсів. Браузери виконують розмітку HTML та інтерпретують її для відображення сторінок користувачам.

3.14. Мова стилів CSS

CSS (Cascading Style Sheets) є мовою стилізації для веб-сторінок. Вона використовується разом з HTML для задання зовнішнього вигляду та оформлення елементів веб-сторінки, таких як кольори, шрифти, розташування, розміри та інші властивості [14].

CSS дозволяє розділити стиль веб-сторінки від її структури. Замість визначення стилів безпосередньо в HTML-коді, вони описуються окремо в CSS-файлах або внутрішніх стилях (embedded styles), що дозволяє зберігати стильові правила централізовано і застосовувати їх до різних сторінок.

CSS використовує правила стилів, що складаються з селекторів та властивостей. Селектори вказують на які елементи HTML будуть застосовані стилі, а властивості визначають конкретні атрибути цих елементів, такі як колір тексту, фоновий колір, розміри, відступи, рамки та інші.

Одна з головних переваг CSS полягає у можливості застосовувати стилі до груп елементів. Замість повторення стилів для кожного елемента окремо, можна використовувати класи, ідентифікатори або елементні селектори для визначення груп елементів, до яких застосовуються однакові стилі.

CSS також підтримує концепцію каскаду, що дозволяє контролювати пріоритетність стилів. Це означає, що якщо декілька стилів конфліктують для одного елемента, то можна використовувати специфічність селекторів або використовувати властивість "!important" для встановлення пріоритету.

3.15. Мова JavaScript

JavaScript - це високорівнева, об'єктно-орієнтована мова програмування, яка використовується для розробки динамічних та інтерактивних веб-додатків. Вона забезпечує можливість створювати функції, які взаємодіють з елементами веб-сторінки, маніпулюють їх властивостями та реагують на події, що виникають під час взаємодії користувача зі сторінкою [15].

JavaScript виконується безпосередньо в браузері користувача, що робить його основною мовою для реалізації клієнтської логіки. Він може взаємодіяти з DOM (Document Object Model) - представленням структури HTML-сторінки, змінювати його елементи, створювати нові, видаляти чи переміщувати їх. JavaScript також дозволяє валідувати дані, виконувати асинхронні запити до сервера, анімувати елементи сторінки, обробляти форми, керувати кукісами та багато іншого.

Одна з ключових особливостей JavaScript полягає в його можливості працювати з об'єктами. Він підтримує об'єктно-орієнтовану парадигму програмування, що дозволяє створювати класи,

об'єкти, використовувати наслідування та інші концепції ООП. JavaScript також надає багато вбудованих об'єктів, таких як String, Array, Date та Math, які спрощують роботу зі строками, масивами, датами та математичними операціями.

Мова JavaScript є мультипарадигмальною, що означає, що вона підтримує не тільки об'єктно-орієнтоване програмування, але й функціональне програмування. Функції можуть бути передані як аргументи, збережені в змінних, повертатися як результат іншої функції, що дозволяє створювати зручні та елегантні рішення.

JavaScript є однією з найпоширеніших мов програмування, яка використовується для розробки веб-додатків. Вона має багато фреймворків та бібліотек, таких як React, Angular і Vue.js, які спрощують процес розробки та покращують продуктивність розробника. JavaScript також використовується для розробки серверної логіки за допомогою платформи Node.js, що дозволяє виконувати JavaScript на стороні сервера.

В цілому, JavaScript є потужним інструментом, який дозволяє створювати динамічні та інтерактивні веб-додатки, забезпечувати взаємодію з користувачем та створювати багатофункціональні веб-сторінки.

3.16. Bootstrap

Bootstrap - це найпопулярніший фреймворк для розробки веб-інтерфейсів. Він надає набір заздалегідь стилізованих компонентів, шаблонів, розмітки та JavaScript-плагінів, які дозволяють розробникам швидко створювати сучасні та адаптивні веб-додатки [11].

Основна перевага Bootstrap полягає в його гнучкості та легкості використання. Він забезпечує однорідний та консистентний вигляд елементів інтерфейсу на різних пристроях та в різних браузерах. Завдяки використанню готових компонентів, розробникам не потрібно витрачати багато часу на написання стилів та скриптів з нуля. Вони можуть просто використовувати готові класи і HTML-структури для створення багатофункціональних елементів, таких як меню, кнопки, форми, каруселі, модальні вікна та багато іншого.

Bootstrap також підтримує адаптивний дизайн, що дозволяє автоматично пристосовувати веб-додатки до різних розмірів екранів, включаючи мобільні пристрої. Це забезпечує оптимальний вигляд та взаємодію з додатком незалежно від пристрою, на якому він відображається.

Крім того, Bootstrap має велику спільноту розробників, яка активно підтримує фреймворк, надає документацію, приклади та відповіді на питання. Це сприяє швидкому вирішенню проблем та підтримці в процесі розробки.

Загалом, Bootstrap є потужним інструментом для розробки сучасних та привабливих веб-додатків. Він дозволяє розробникам ефективно використовувати час та зусилля, прискорюючи процес розробки та забезпечуючи професійний вигляд інтерфейсу.

Розділ 3. Розробка вебсайту. Створення функціоналу застосунку.

4. Розпочнемо з створення простого веб-застосунку на Flask

```
from flask import Flask

app = Flask(__name__)

if __name__ == '__main__':

        app.run(debug=True)
```

Загалом, цей код створює об'єкт додатку Flask і запускає його на локальному сервері у режимі налагодження. Проте, без додавання маршрутів (routes) та функціоналу, додаток не буде мати жодного видимого вмісту або дій. Його можна використовувати як основу для подальшого розширення функціоналу та розробки веб-застосунків на основі Flask.

4.1. Імплементуємо API для відображення погоди у місті Львів

Для цього спочатку зареєструємось на сайті https://www.worldweatheronline.com/ - є веб-сайтом, який надає погодні прогнози та метеорологічні дані з усього світу. Основна функція сайту - це надання актуальних та прогнозних погодних умов, таких як температура, вологість, швидкість вітру, опади та багато іншого.

Там отримаємо тимчасовий ключ API і перевіримо його роботу на тому ж сайті,

Запит до сайту https://www.worldweatheronline.com/ для отримання погодніх даних зазвичай використовує ключ API та параметри, що вказуються у URL. Для прикладу, запит може мати такий вигляд:

https://api.worldweatheronline.com/premium/v1/weather.ashx?key=YOUR_API_KEY&q=Lviv,Ukraine&format=json

Після відправки запиту до сайту з цим URL, ви отримаєте відповідь, яка міститиме погодні дані

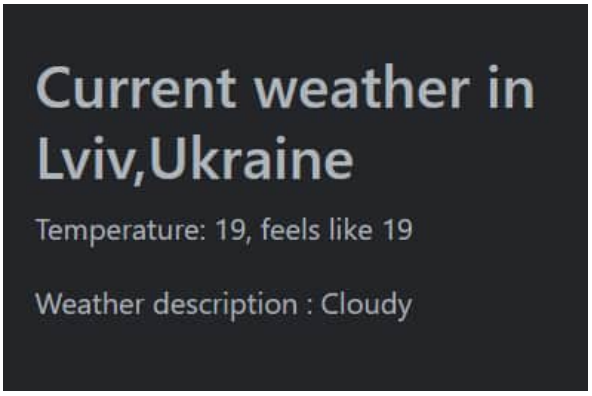


Рис.1. Готовий оброблений запит до worldweatheronline в фінальній версії веб-застосунку

у форматі JSON. Обробимо ці дані в своєму коді, щоб відобразити їх в нашому веб-застосунку Flask, наприклад, за допомогою шаблонів та даних, переданих в контекст шаблону.

Важливо пам'ятати, що ключ API є конфіденційним і не повинен бути розголошений або розповсюджуваний.

```python
import requests

from flask import current_app


def weather_by_city(city_name):
    weather_url = current_app.config['WEATHER_URL']
    params = {
        'key' : current_app.config['WEATHER_API_KEY'],
        'q' : city_name,
        "format" : "json",
        "num_of_days" : 1,
        "lang" : "uk"
    }
    try:
        result = requests.get(weather_url, params=params)
        result.raise_for_status()
        weather = result.json()
        if 'data' in weather:
            if "current_condition" in weather['data']:
                try:
                    return weather['data']['current_condition'][0]
                except(IndexError, TypeError):
                    return False
    except(requests.RequestException, ValueError):
        print('Net error')
        return False
    return False

if __name__ == '__main__':
    w = current_app.config['WEATHER_API_KEY']
    print(w)
```

Рис.2. Код запиту та подальшої обробки отриманих даних

Також цей код реалізує механізм обробки помилок, який дозволяє виявляти та обробляти проблеми під час отримання погодних даних. При виникненні помилки виконання запиту або обробки даних, функція поверне False як результат, на сторінку замість інформації про погоду виведеться повідомлення Net error в відповідному для погоди блоці.

```
<div class="col-4">
    <h2>Current weather in Lviv,Ukraine</h2>
    {% if weather %}
        <p>Temperature: {{ weather.temp_C }}, feels like {{ weather.FeelsLikeC }}</p>
        <p>Weather description : {{ weather.weatherDesc[0]['value'] }}</p>
    {% else %}
        Weather forecast temporarly unavailible
    {% endif %}
</div>
```

Рис.3. Застосування отриманої інформації в HTML темплейті для виведення на сторінку додатку

4.2. Застосуємо фреймворк Bootstrap

Одразу підключимо фреймворк Bootstrap до наших HTML темплейтів. Для цього використаємо посилання на CDN (Content Delivery Network).

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.rtl.min.cs
<link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
```

Рис.4. Приблизний вигляд CDN

Використовуючи функціонал Bootstrap форматуємо вигляд нашого веб-додатку, як видно на Рис.1. фоновий колір сайту поміняний на темний, що є одною з можливостей фреймворку і застосовується кодом data-bs-theme="dark", на Рис.2. блок має клас col-4 що є одним з класів Bootstrap котрий займає 4 колонки у сітці Bootstrap. Це дозволяє розмістити його на сторінці відповідно до сітки, що забезпечує респонсивний дизайн. В подальшому за допомогою фреймворку буде створена навігаційна панель, всі кнопки та форми веб-застосунку.

4.3. Реалізуємо функцію збору новин з стороннього сайту

Процес збору новин реалізуємо за допомогою модулю BeautifulSoup, він буде полягати в тому:

- Ми будемо робити запит по посиланню https://dou.ua/lenta/tags/python/?lang=en (котре веде нас до всіх публікацій, що стосується новин пов’язаних з мовою Python)
- Створюється об'єкт BeautifulSoup з отриманого HTML-коду, використовуючи парсер 'html.parser'.
- Знаходимо всі новини на сторінці, шукаючи елемент з класом 'b-lenta', а всередині нього знаходимо всі елементи <article> з класом 'b-postcard'.
- Ітеруємося по кожній новині і отримуємо її заголовок, URL та дату публікації.
- Заголовок новини отримується з елемента <h2> з класом 'title', а URL - з атрибута href елемента <a> всередині <h2>. Обидва значення очищаються від зайвих пробілів методом strip().

- Дата публікації отримується з елемента <time> з класом 'date'. Спочатку спробуємо виконати парсинг дати в форматі 'день місяць, година:хвилина', де рік замінюється на поточний рік. Якщо цей формат не відповідає, тоді парсимо дату в форматі 'день місяць рік, година:хвилина'.

```python
def get_news_snippets():
    html = get_html("https://dou.ua/lenta/tags/python/?lang=en")
    if html:
        soup = BeautifulSoup(html, 'html.parser')
        all_news = soup.find('div', class_='b-lenta').findAll('article', class_='b-postcard')
        result_news = []
        for news in all_news:
            find_title = news.find('h2', class_='title').find('a')
            title = find_title.text.strip()
            url = news.find('h2', class_='title').find('a')['href']
            published = news.find('time', class_='date').text

            try:
                published = datetime.strptime(published, '%d %B, %H:%M').replace(year=datetime.now().year)
            except ValueError:
                published = datetime.strptime(published, '%d %B %Y, %H:%M')
            save_news(title, url, published)
```

Рис.5. Парсинг назв, дати публікацій по запиту

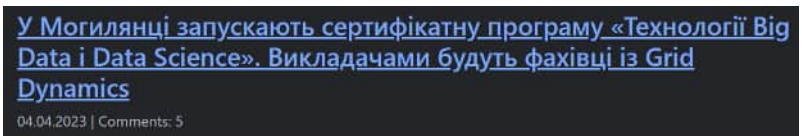


Рис.6. Приклад вигляду блоку з новиною на сторінці додатку

```
<div class ="col-8">
    {% include('messages.html') %}
    <h2>News</h2>
    {% for news in news_list %}
    <h3><a href="{{ url_for('news.single_news', news_id=news.id) }}">{{ news.title }}</a></h3>
    <p>{{ news.published.strftime('%d.%m.%Y') }} | Comments: {{ news.comments_count() }}</p>
    <hr />
    {% endfor %}

</div>
```

Рис.7. Застосування отриманої інформації в HTML темплейті для виведення на сторінку додатку

Як видно на Рис.6 окрім назви публікації, також виводиться дата в зрозумілому форматі та поточну кількість коментарів під цією новиною, реалізація котрих буде описана пізніше. Також по Рис.6. можна зрозуміти, що назва публікації є посиланням.

4.4. Реалізація збору контенту з отриманих раніше публікацій

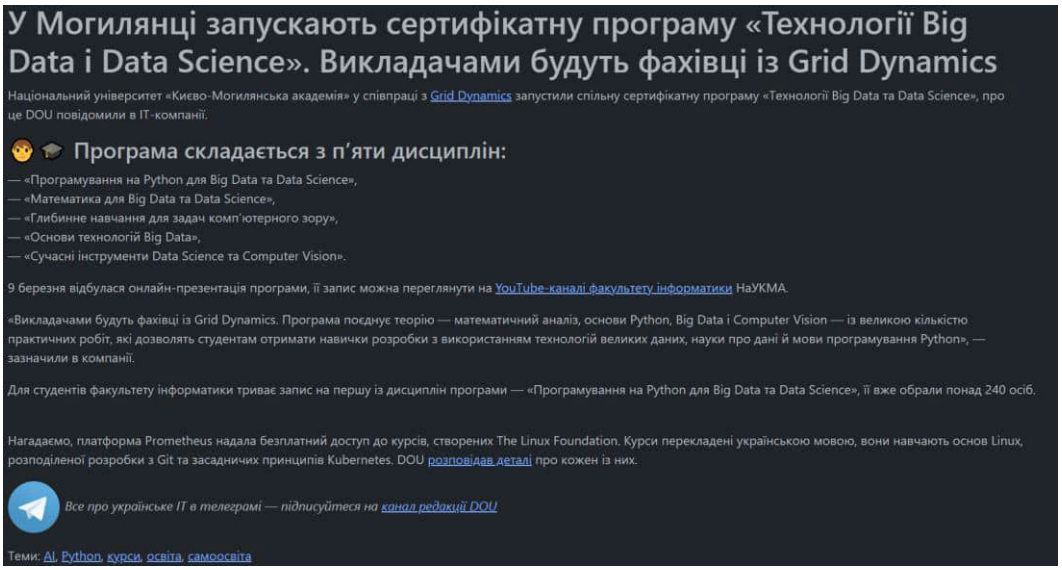
- Для кожної новини, отриманої з попереднього пункту, виконується наступний код.
- Виконується функція get_html(news.url), яка отримує HTML-код сторінки, що відповідає URL новини.
- Перевіряється, чи був успішно отриманий HTML-код. Якщо так, продовжуємо виконання коду, в іншому випадку пропускаємо наступні кроки для цієї новини.

- Створюється об'єкт BeautifulSoup з отриманого HTML-коду, використовуючи парсер 'html.parser'.
- Знаходимо елемент <article> з класом 'b-typo b-typo_post', який містить текст новини.
- Використовуючи метод decode_contents(), отримуємо вміст елементу <article> у вигляді рядка тексту.
- Перевіряємо, чи отриманий текст новини не є порожнім. Якщо так, присвоюємо його полю text об'єкта новини.

Процес повторюється для наступної новини зі списку новин без тексту.

```
def get_news_content():
    news_without_text = News.query.filter(News.text.is_(None))
    for news in news_without_text:
        html = get_html(news.url)
        if html:
            soup = BeautifulSoup(html, 'html.parser')
            news_text = soup.find('article', class_='b-typo b-typo_post').decode_contents()
            if news_text:
                news.text = news_text
                db.session.add(news)
                db.session.commit()
```

Рис.8. Код збору контенту новин



У Могилянці запускають сертифікатну програму «Технології Big Data і Data Science». Викладачами будуть фахівці із Grid Dynamics

Національний університет «Києво-Могилянська академія» у співпраці з Grid Dynamics запустили спільну сертифікатну програму «Технології Big Data та Data Science», про це DOU повідомили в IT-компанії.

Програма складається з п'яти дисциплін:

— «Програмування на Python для Big Data та Data Science»,
— «Математика для Big Data та Data Science»,
— «Глибинне навчання для задач комп'ютерного зору»,
— «Основи технологій Big Data»,
— «Сучасні інструменти Data Science та Computer Vision».

9 березня відбулася онлайн-презентація програми, її запис можна переглянути на YouTube-каналі факультету інформатики НаУКМА.

«Викладачами будуть фахівці із Grid Dynamics. Програма поєднує теорію — математичний аналіз, основи Python, Big Data і Computer Vision — із великою кількістю практичних робіт, які дозволять студентам отримати навички розробки з використанням технологій великих даних, науки про дані й мови програмування Python», — зазначили в компанії.

Для студентів факультету інформатики триває запис на першу із дисциплін програми — «Програмування на Python для Big Data та Data Science», її вже обрали понад 240 осіб.

Нагадаємо, платформа Prometheus надала безплатний доступ до курсів, створених The Linux Foundation. Курси перекладені українською мовою, вони навчають основ Linux, розподіленої розробки з Git та засадничих принципів Kubernetes. DOU розповідав деталі про кожен із них.

Все про українське IT в телеграмі — підписуйтеся на канал редакції DOU

Теми: AI, Python, курси, освіта, самоосвіта

Рис.9. Приклад контенту новини в додатку

На Рис.8. також можна побачити 2 строчки коду db.session.add(news) та db.session.commit() котрі відповідають за збереження отриманих даних в базу даних, роботу котрої буде описано нижче.

```
<div class="row">
    <div class =" news-content">
        {% include('messages.html') %}
        {{ news.text|safe }}
        {% if news.comments %}
            <h3>Comments:</h3>
            {% for comment in news.comments %}
            <div class="card" style="width: 18rem;">
                <div class="card-body">
                <p class="card-text">{{ comment.text }}</p>
                <p class="card-text">Published: {{ comment.created.strftime('%d.%m.%Y') }} ||
                    User: {{ comment.user.username }}</p>
                </div>
            </div>
            {% endfor %}
        {% endif %}
        {% if current_user.is_authenticated %}
            <form action="{{ url_for('news.add_comment') }}" method="POST" class="comm">
                {{ comment_form.hidden_tag() }}
                <div class="form-group">
                    {{ comment_form.comment_text.label }}
                    {{ comment_form.comment_text() }}
                </div>
                <div class="form-group">
                {{ comment_form.submit() }}
                </div>
            </form>
        {% else %}
            <p>
                <a href="{{ url_for('user.login', next=request.full_path) }}">Sign up</a> to send comments
            </p>
        {% endif %}
    </div>
```

Рис.10. Код що оброблює контент новин та відобажає їх в HTML темлпейті

4.5. Робота з базою даних

Для роботи з базами даних в проекті було обрано плагін SQLAlchemy через те, що для роботи з ним не потребуються навички в SQL, в разі чого він забезпечує легкий перехід з sqlite на інші СУБД та також плагін забезпечує захист запитів до бази даних від SQL-інєкцій.

Для проекту була обрана база даних SQlite через те, що вона проста в налаштуванні і роботі, та також її легко розвертати і підтримувати. Хоча вона і має вагомі мінуси такі, як невисока надійність, погана витривалість навантажень та мало вбудованих інструментів.

Отже, для роботи з базою даних спочатку було встановлено бібліотеку Flask-SQLAlchemy, котра робить роботу з SQLAlchemy з Flask-застосунку значно зручнішою

```
import os

basedir = os.path.abspath(os.path.dirname(__file__))
SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(basedir, '..', 'webapp.db')
```

Рис.11. Початкова конфігурація sqlite-бази.

На Рис.11. задано шлях до нашої sqlite-бази даних і присвоєно їй константу SQLALCHEMY_DATABASE_URI для того, що в майбутньому при звертанні до бази даних не було б потрібно кожен раз прописувати повний шлях та щоб прибрати з основного коду приватні дані проекту.

Для перегляду структури та даних в базі було встановлено DB Browser for SQLite, котрий надає зручний графічний інтерфейс.

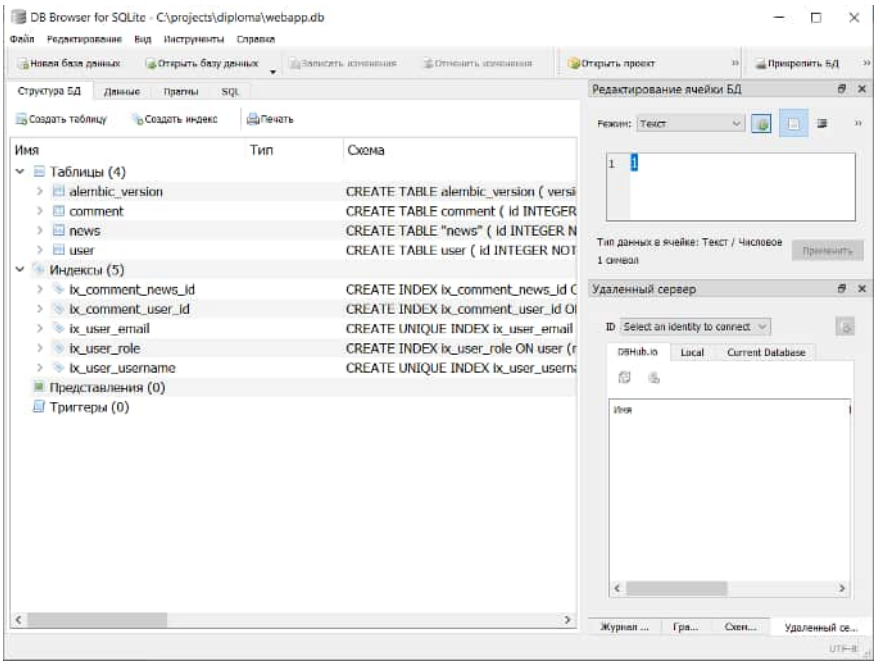


Рис.12. Інтерфейс програми DB Browser

4.6. Створення моделей в базі даних

Створимо базу даних за допомогою коду, для цього створимо файл create_db.py з Рис.13. та запустимо його, SQLALchemy сама створить файл бази даних і відповідні таблиці задані кодом, як на Рис.14.

```
from webapp import db, create_app

db.create_all(app=create_app())
```

Рис.13. create_db.py

Створення моделей в базі даних відбувається за допомогою коду

```python
from datetime import datetime
from sqlalchemy.orm import relationship
from webapp.db import db

class News(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String, nullable=False)
    url = db.Column(db.String, unique=True, nullable=False)
    published = db.Column(db.DateTime, nullable=False)
    text = db.Column(db.Text, nullable=True)

    def comments_count(self):
        return Comment.query.filter(Comment.news_id == self.id).count()

    def __repr__(self):
        return 'News {} {}'.format(self.title, self.url)
```

Рис.14. Опис моделі News

news		CREATE TABLE "news" (id INTEGER NOT NULL, title '
id	INTEGER	"id" INTEGER NOT NULL
title	VARCHAR	"title" VARCHAR NOT NULL
url	VARCHAR	"url" VARCHAR NOT NULL
published	DATETIME	"published" DATETIME NOT NULL
text	TEXT	"text" TEXT

Рис.15. Таблиця news в DB Browser

Запис новин тепер відбувається одразу в базу даних, а з неї витягується в наш веб-додаток, що суттєво економить ресурси, бо надає можливість використовувати одній й ті самі дані безліч разів

На даному етапі для збору новин потребується запуск файлу get_all_news.py з кодом з Рис.5. та Рис.8., але далі буде реалізований автоматичний збір новин по таймеру за допомогою WSL, Redis та Celery

4.7. Реалізація авторизації та перевірки прав користувача

За допомогою Flask-WTF створимо форми логіну та реєстрації

```python
class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()], render_kw={"class": "form-control"})
    password = PasswordField('Password', validators=[DataRequired()], render_kw={"class": "form-control"})
    remember_me = BooleanField('Remember me', default=True, render_kw={"class": "form-check-input"})
    submit = SubmitField('Submit', render_kw={'class': "btn btn-primary"})


class RegistrationForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()], render_kw={"class": "form-control"})
    email = StringField('User email', validators=[DataRequired(), Email()], render_kw={"class": "form-control"})
    password2 = PasswordField('Repeat password', validators=[DataRequired(), EqualTo('password')], render_kw={"class": "form-control"})
    password = PasswordField('Enter password', validators=[DataRequired()], render_kw={"class": "form-control"})
    submit = SubmitField('Submit', render_kw={'class': "btn btn-primary"})
```

Рис.16. Форми для логіну та рєстрації за допомогою Flask-WTF

Код містить два класи форм, що використовуються для створення форм для входу і реєстрації користувачів.

Клас LoginForm містить поля для введення імені користувача, пароля та прапорця для запам'ятовування користувача.

RegistrationForm містить поля для введення імені користувача, електронної пошти та пароля, який користувач повинен ввести двічі для перевірки правильності.

```
<h1><div class="mx-auto p-2" style="width: 300px;">{{ page_title }}</div></h1>
<div class="row">
    <div class="mx-auto p-2" style="width: 300px;">
        {% include('messages.html') %}
        <form action="{{ url_for('user.process_login') }}" method="POST">
            {{ form.hidden_tag() }}
            <div class="mb-3">
                {{ form.username.label }}
                {{ form.username() }}
            </div>
            <div class="mb-3">
                {{ form.password.label }}
                {{ form.password() }}
            </div>
            <div class="form-group form-check">
                {{ form.remember_me() }}
                {{ form.remember_me.label(class_='form-check-label') }}
            </div>
            {{ form.submit() }}
        </form>
    </div>
```

Рис.17. Застосування форми логіну в HTML

```
<h1><div class="mx-auto p-2" style="width: 300px;">{{ page_title }}</div></h1>
{% include('messages.html') %}
<div class="row" >
    <div class="mx-auto p-2" style="width: 300px;">
        <form action="{{ url_for('user.process_reg') }}" method="POST">
            {{ form.hidden_tag() }}
            <div class="mb-3">
                {{ form.username.label }}
                {{ form.username() }}
            </div>
            <div class="mb-3">
                {{ form.email.label }}
                {{ form.email() }}
            </div>
            <div class="mb-3">
                {{ form.password.label }}
                {{ form.password() }}
            </div>
            <div class="mb-3">
                {{ form.password2.label }}
                {{ form.password2() }}
            </div>
            {{ form.submit() }}
        </form>
    </div>
```

Рис.18. Застосування форми реєстрації в HTML

Flask-WTF дозволяє відрендерити форми ще в Python коді та застосовувати їх як переміні в HTML коді, також надає швидкий доступ до даних з форм, та має функції валідації даних, наприклад як на Рис.16., де в RegistrationForm прописані валідатори для паролів, котрий перевіряє чи повторний пароль дорівнює першому.

Також Flask-WTF забезпечує захист від різних типів атак, таких як атаки типу "Cross-Site Request Forgery" (CSRF).

Захист CSRF включений за замовчуванням і автоматично застосовується до всіх форм.

В класі RegistrationForm також було оброблено валідацію імені користувача та пошти користувача, що було неможливо створювати 2 чи більше акаунтів з однаковим іменем користувача або пошти, в разі не проходження валідації користувачу буде показана та чи інша помилка з Рис.19.

```python
def validate_username(self, username):
    user_count = User.query.filter_by(username=username.data).count()
    if user_count > 0:
        raise ValidationError('User with this username already exists')

def validate_email(self, email):
    user_count = User.query.filter_by(email=email.data).count()
    if user_count > 0:
        raise ValidationError('User with this email already exists')
```

Рис.19. Валідація імені користувача та пошти

Демонстрація форм в запущеному додатку:

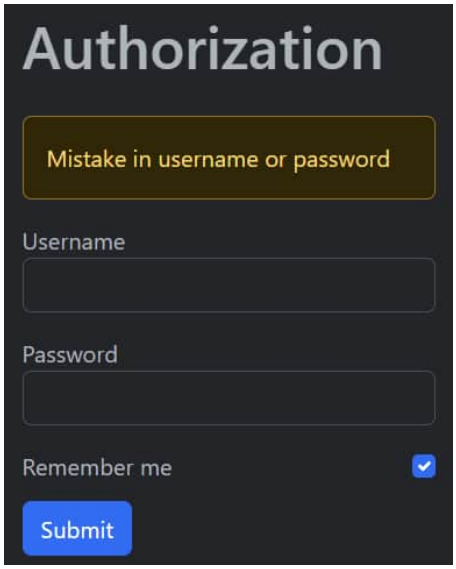


Рис.20. Форма логіну після неправильного введеня паролю або імені користувача

На Рис.20. видно вже оброблену за допомогою Flask-WTF та Bootstrap форму з характерими полями та повідомленням про помилку у введені паролю або юзернейму.

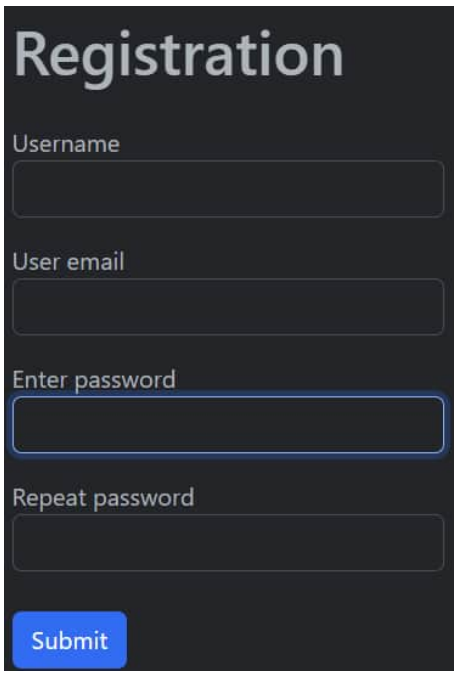


Рис.21. Форма реєстрації

Форма реєстрації на Рис.21., також створена за допомогою Flask-WTF та Bootstrap, має описані вище можливості валідації.

Обидві форми відправляють та беруть дані з бази даних, для цього була створена модель User з потрібними полями. Процес створення аналогічний процесу описаному в пункті 4.7.

Також для додаткового захисту даних було додано хешування паролів за допомогою werzeug, цей додатковий захист не доволить вкрасти паролі користувачів з бази даних, так як для розхешованню потрібен унікальний ключ, котрий відсутній в відкритому доступі.

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(48), index=True, unique=True)
    password = db.Column(db.String(128))
    role = db.Column(db.String(10), index=True)
    email = db.Column(db.String(50),index=True, unique=True)
```

Рис.22. Опис моделі User

```
def set_password(self, password):
    self.password = generate_password_hash(password)

def check_password(self, password):
    return check_password_hash(self.password, password)
```

Рис.23. Хешування паролів

В проекті також є реалізація перевірки прав користувача та наділення їх особливими можливостями, в нашому проекті це несе лише демонстраційний характер, так користувач з

роллю admin, буде мати можливість переходити на додаткову сторінку, а звичайний користувач не зможе на неї попасти.

Перевірка ролі відбувається за допомогою коду з Рис.24., роль встановлюється вручну

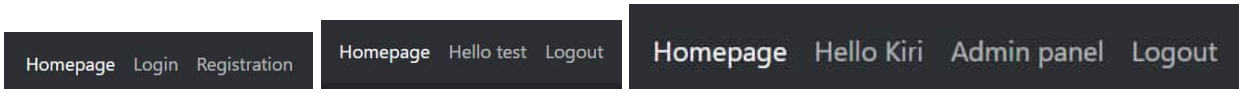
```
@property
def is_admin(self):
    return self.role == 'admin'

def __repr__(self):
    return '<User {}>'.format(self.username)
```

Рис.24. Перевірка ролі користувача

За допомогою реєстрації, логіну та перевірки ролі користувача можливо налаштувати функціонал під кожного з 3 можливих видів користувачів, так незареєстрований користувач не зможе в подальшому лишати коментарів під новинами, а адмін буде мати додаткову сторінку в веб додатку заблоковану для інших користувачів

Ось порівняння панелі незалогіненого користувача, звичайного юзера та адміна, адмін панель доступна ЛИШЕ адмінам та не може бути відкрита без відповідної ролі.



Панелі юзері

У випадку спроби звичайного або незареєстрованого користувача перейти по http://127.0.0.1:5000/admin/ їх буде редіректнуто на головну сторінку та показано відповідну помилку з Рис.25. Даний функціонал описаний кодом в файлах:

webapp\user\views.py,

webapp\templates\menu.html,

webapp\admin\views.py,

webapp\templates\admin\index.html

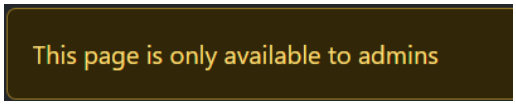


Рис.25.

```python
@blueprint.route('/')
@admin_required
def admin_index():
        title = "Control panel"
        return render_template('admin/index.html', page_title=title)
```

Рис.26. Додатковий функціонал адміна

4.8. Функціонал запам’ятовування користувача

Якщо користувач авторизується і закриє браузер, тоді при наступному вході йому прийдеться проходити авторизацію повторно. Для уникнення цього було реалізовано функціонал запам’ятовування користувача по бажанню.

На Рис.16. в класі LoginForm створено булеву форму, що називається rember_me має дефолтний параметр True і виглядає, як галочка з Рис.20.

На Рис.27. в функції process_login реалізуємо перевірку чи була проставлена галка в формі

```python
@blueprint.route('/process-login', methods=['POST'])
def process_login():
    form = LoginForm()

    if form.validate_on_submit():
        user = User.query.filter(User.username == form.username.data).first()
        if user and user.check_password(form.password.data):
            login_user(user, remember=form.remember_me.data)
            flash("You are logged in")
            return redirect(url_for('news.index'))

    flash('Mistake in username or password')
    return redirect(url_for('user.login'))
```

Рис.27.

Тривалість запам’ятовування налаштовується через конфігурацію Flask, в нашому випадку буде виставлено 5 днів тривалості зберігання логіну користувача :

```python
from datetime import timedelta
REMEMBER_COOKIE_DURATION = timedelta(days=5)
```

Рис.28. Конфігурація тривалості

4.9. Розбиття проекту на модулі та використання Blueprint

Blueprint - це стандартний метод поділу Flask-додатків на модулі. Він надає зручний спосіб розділення великого додатку на невеликі функціональні модулі та підтримку структури URL з іменами.

Використовуючи Blueprint, буде організовано наш додаток на незалежні модулі, які виконують конкретні функції. Кожен модуль має свої власні маршрути, шаблони, статичні файли і логіку

обробки запитів. Це дозволяє підтримувати чистоту коду та покращує зрозумілість структури вашого додатку.

Blueprint також надає можливість іменування URL для кожного модуля. Вкажемо префікси URL для Blueprint, що дозволяє створити ієрархію URL з використанням іменованих модулів. Це зробить структуру нашого додатку більш організованою та зручною для обробки URL-адрес.

Модулі, на які було розбито наш проект:

User – авторизація, реєстрація, профіль користувача

News – сторінка новин, збір новин, коментарі

Admin – приклад адмін панелі

```
blueprint= Blueprint('user', __name__, url_prefix='/users')
```

Рис.29. Створення блюпрінту для модуля User

```
def create_app():
    app = Flask(__name__)
    app.config.from_pyfile('config.py')
    db.init_app(app)
    migrate = Migrate(app,db)

    login_manger = LoginManager()
    login_manger.init_app(app)
    login_manger.login_view = 'user.login'
    app.register_blueprint(admin_blueprint)
    app.register_blueprint(user_blueprint)
    app.register_blueprint(news_blueprint)

    @login_manger.user_loader
    def load_user(user_id):
        return User.query.get(user_id)

    return app
```

Рис.30. Файл __init__.py після рефакторингу

На Рис.30. продемонстрований файл __init__.py котрий відповідає за запуск Flask-додатку, як можна побачити функція create_app() створює Flask додаток з конфігурацією з файлу config.py, підключує базу даних SQLAlchemy та міграції з Flask-Migrate.

Далі, ініціалізує об'єкт LoginManager для авторизації користувачів та зареєстровує блюпрінти, які містять маршрути та логіку відповідної частини додатку. Також визначає функцію load_user(), яка повертає об'єкт користувача на основі ідентифікатора користувача.

Кожен з цих модулів має свій Blueprint, свої власні маршрути, статичні файли такі як views.py, models.py для відповідної роботи з користувачем або базою даних. Така організація файлів

суттєво покращує чистоту коду, дозволяє створити зрозумілу сторонньому розробнику ієрархію проекту.

Отже маємо в результаті маємо дану структуру папок після застосування Blueprint та рефакторингу коду зображену на Рис.31.

Проект поділений на модулі, є окрема папка для html темплейтів, де в свою чергу все також поділено функціоналу того чи іншого темплейта.

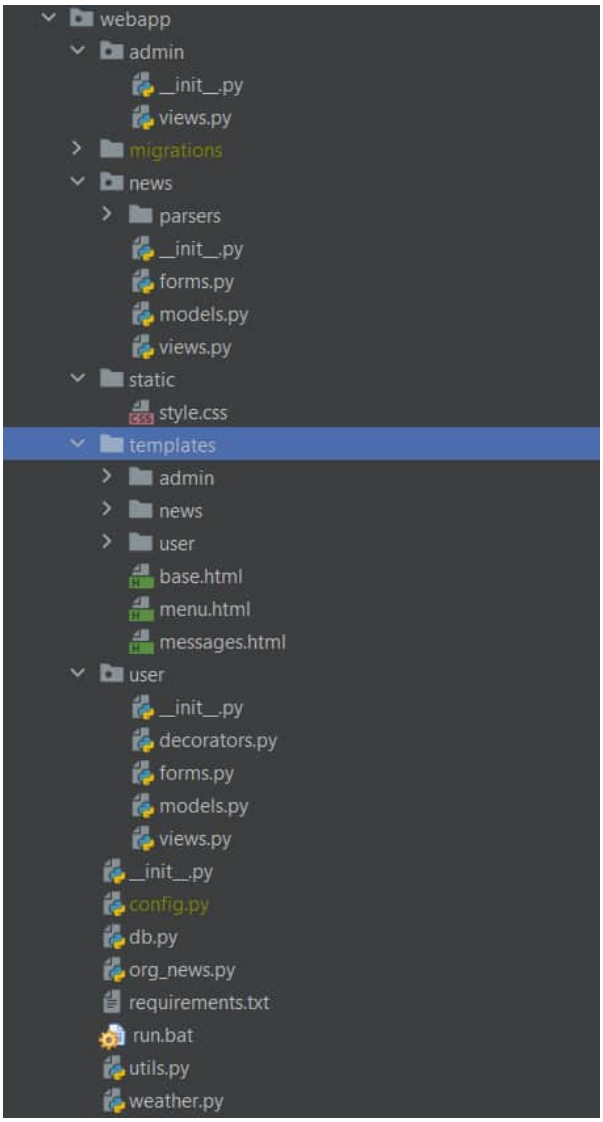


Рис.31. Приблизна структура проекту

4.10. Скрипт для запуску додатку

На початковому етапі проект запускався набором команд в терміналі середовища розробки, що є не надто зручно. Для полегшення та пришвидшення запуску додатку було створено bat файл з набором команд потрібних для запуску

Рис.32. Вміст файлу run.bat

Команди set FLASK_APP=__init__, set FLASK_ENV='development', і set FLASK_DEBUG=1 використовуються для налаштування змінних середовища для запуску Flask-додатку за допомогою команди flask run.

FLASK_APP=__init__ вказує, що файл __init__.py у вашому основному каталозі проекту є головним файлом додатку Flask.

FLASK_ENV='development' встановлює режим розробки для Flask. У режимі розробки додаток буде перезапускатись автоматично при зміні файлів додатку, а також забезпечує додаткову інформацію про помилки і винятки.

FLASK_DEBUG=1 вмикає режим налагодження (debug mode) для Flask. У цьому режимі будуть виводитись докладні повідомлення про помилки та винятки, що полегшує відлагодження додатку.

Після встановлення змінних середовища, команда flask run запускає сервер розробки Flask, який слухатиме на локальному сервері із використанням порту за замовчуванням (зазвичай 5000). Після запуску серверу ви зможете відкрити додаток у веб-браузері, перейшовши за адресою http://localhost:5000.

4.11. Спеціальний файл __init__.py

Файл з назвою __init__.py з двома підкресленими символами на початку і в кінці назви є спеціальним файлом в Python. Такий файл називається "магічним" або "спеціальним" файлом.

У контексті Flask, файл __init__.py використовується як головний файл додатку. При наявності цього файлу в каталозі, Python розпізнає його як пакет і дозволяє імпортувати його модулі та об'єкти.

При використанні Flask Blueprint, файл __init__.py може містити код для створення та налаштування Blueprint, додавання маршрутів та інших функцій, які визначають поведінку додатку.

4.12. Автоматизація збору новин, Redis&Celery

Для повної автоматизації збору новин було встановлено Redis, просту базу даних типу ключ-значення, яка використовується Celery для зберігання черги задач. Однак, на ОС Windows доступна лише стара версія Redis, тому встановлено Linux-підсистему для Windows, щоб уникнути цих обмежень і спростити роботу з Celery.

```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/kiri/.hushlogin file.
kiri@DESKTOP-ALH821A:/mnt/c/Windows/system32$
```

Рис.33.Консоль підсистеми

Після успішного встановлення WSL, Redis та Celery з підтримкою Redis, переходимо до написання таску для Celery. Таск - це звичайна функція, яку

обгортаємо декоратором celery.task. Її можна викликати напряму або передати на виконання Celery за допомогою методу delay().

```python
flask_app = create_app()
celery_app = Celery('task', broker='redis://localhost:6379/0')

@celery_app.task
def dou_snippets():
    with flask_app.app_context():
        dou.get_news_snippets()

@celery_app.task
def dou_content():
    with flask_app.app_context():
        dou.get_news_content()

@celery_app.on_after_configure.connect
def setup_periodic_tasks(sender, **kwargs):
    sender.add_periodic_task(crontab(minute='*/1'), dou_snippets.s())
    sender.add_periodic_task(crontab(minute='*/1'), dou_content.s())
```

Рис.34. Таск для збору новин

У вказаному коді створюється Flask-додаток і об'єкт Celery. Останній дозволяє запускати задачі для отримання коротких та повних новин від DOU.ua за допомогою періодичної задачі, виконуваної за допомогою crontab.

Отже, була створена повністю автоматизовану перевірку сайту DOU.ua і збір новин в базу даних, якщо є нові записи.

З цим підходом ви зможете регулярно отримувати свіжі новини з DOU.ua безпосередньо у вашу базу даних, завдяки використанню Redis та Celery.

Для запуску задач по розкладу, потребується запустити celery-beat. Саме beat буде слідкувати за розкладом і відправляти задачі worker-ам. Для великих проектів Beat бажано запускати в окремому вікні термінала.

Але для нашого проекту буде достатньо запустити celery з налаштуваннями з Рис. 35.

```
kiri@DESKTOP-ALH821A:/mnt/c/projects/diploma$ sudo service redis-server start
Starting redis-server: redis-server.
kiri@DESKTOP-ALH821A:/mnt/c/projects/diploma$ celery -A tasks worker -B --loglevel=info

 -------------- celery@DESKTOP-ALH821A v5.2.7 (dawn-chorus)
--- ***** -----
-- ******* ---- Linux-5.15.90.1-microsoft-standard-WSL2-x86_64-with-glibc2.35 2023-05-24 11:44:52
- *** --- * ---
- ** ---------- [config]
- ** ---------- .> app:         task:0x7fe956b97490
- ** ---------- .> transport:   redis://localhost:6379/0
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 12 (prefork)
-- ******* ---- .> task events: OFF (enable -E to monitor tasks in this worker)
--- ***** -----
 -------------- [queues]
                .> celery           exchange=celery(direct) key=celery
```

Рис.35. Запуск Redis та Celery в WSL

Робота таску виглядає ось так:

```
[2023-05-24 11:44:52,742: INFO/Beat] beat: Starting...
[2023-05-24 11:44:52,777: INFO/Beat] Scheduler: Sending due task tasks.dou_snippets() (tasks.dou_snippets)
[2023-05-24 11:44:52,782: INFO/Beat] Scheduler: Sending due task tasks.dou_content() (tasks.dou_content)
[2023-05-24 11:44:53,429: INFO/MainProcess] Connected to redis://localhost:6379/0
[2023-05-24 11:44:53,431: INFO/MainProcess] mingle: searching for neighbors
[2023-05-24 11:44:54,437: INFO/MainProcess] mingle: all alone
[2023-05-24 11:44:54,445: INFO/MainProcess] celery@DESKTOP-ALH821A ready.
[2023-05-24 11:44:54,447: INFO/MainProcess] Task tasks.dou_snippets[097883a0-b57c-4596-bff6-c0f5e14713e6] received
[2023-05-24 11:44:54,448: INFO/MainProcess] Task tasks.dou_content[45b029f4-cd6c-48a5-8a1f-3c7759d2449c] received
[2023-05-24 11:44:54,489: INFO/ForkPoolWorker-9] Task tasks.dou_content[45b029f4-cd6c-48a5-8a1f-3c7759d2449c] succeeded i
n 0.040083026000000491s: None
[2023-05-24 11:44:55,273: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,276: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,278: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,281: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,283: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,286: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,289: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,291: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,295: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,298: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,302: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,304: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,307: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,310: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,313: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,315: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,318: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,320: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,323: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,325: WARNING/ForkPoolWorker-8] 1
[2023-05-24 11:44:55,326: INFO/ForkPoolWorker-8] Task tasks.dou_snippets[097883a0-b57c-4596-bff6-c0f5e14713e6] succeeded
in 0.8772336239999987s: None
[2023-05-24 11:45:00,000: INFO/Beat] Scheduler: Sending due task tasks.dou_content() (tasks.dou_content)
[2023-05-24 11:45:00,001: INFO/Beat] Scheduler: Sending due task tasks.dou_snippets() (tasks.dou_snippets)
[2023-05-24 11:45:00,001: INFO/MainProcess] Task tasks.dou_content[8791f1c6-fd72-4f44-ade0-64106059ab5c] received
[2023-05-24 11:45:00,002: INFO/MainProcess] Task tasks.dou_snippets[af3b4b5f-7b92-4119-a4f2-3a68507eefd0] received
[2023-05-24 11:45:00,017: INFO/ForkPoolWorker-8] Task tasks.dou_content[8791f1c6-fd72-4f44-ade0-64106059ab5c] succeeded i
n 0.014239989000003561s: None
```

Рис.36. Робота таску при наявності новин в базі даних

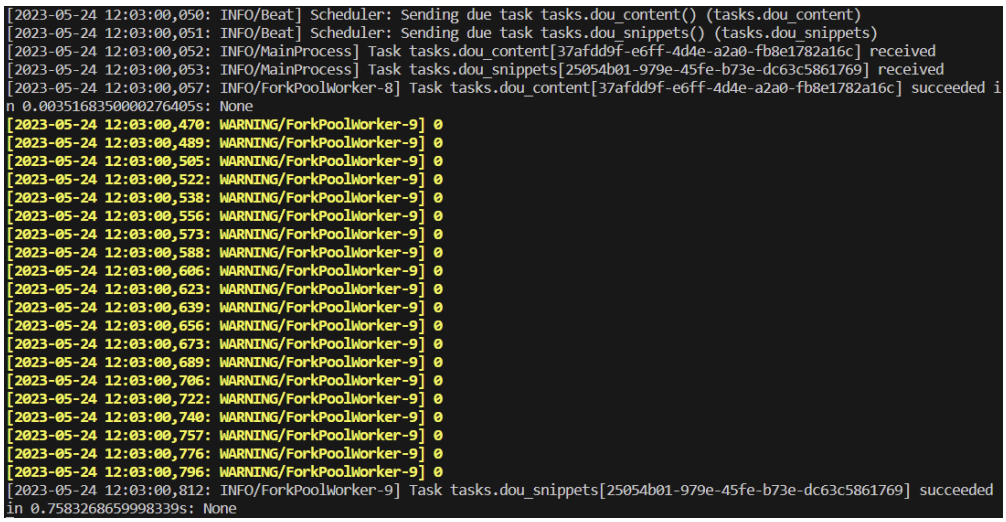


Рис.37.Робота таску при відсутності новин в базі даних

4.13. Реалізація функціоналу коментарів

Для реалізації функціоналу коментування новин і збереження коментарів в базі даних, треба ускладнити структуру нашої бази даних. Кожен коментар повинен належати конкретному користувачеві та бути прив'язаним до певної новини.

Це можна досягти за допомогою полів типу ForeignKey та механізму relationship з SQLAlchemy.

Створимо модель коментарів

```python
class Comment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    text = db.Column(db.Text, nullable=False)
    created = db.Column(db.DateTime, nullable=False, default=datetime.now())
    news_id = db.Column(
        db.Integer,
        db.ForeignKey('news.id', ondelete='CASCADE'),
        index=True
    )
    user_id = db.Column(
        db.Integer,
        db.ForeignKey('user.id', ondelete='CASCADE'),
        index=True
    )
    news = relationship('News', backref='comments')
    user = relationship('User', backref='comments')

    def __repr__(self):
        return '<Comment {}>'.format(self.id)
```

Рис.38. Модель коментарів

У моделі Comment ви можете використовувати поле ForeignKey для встановлення зв'язку з таблицею користувачів (User) та поле ForeignKey для встановлення зв'язку з таблицею новин (News). За допомогою механізму relationship ви можете визначити зв'язок між моделями та

отримувати доступ до коментарів, які належать конкретному користувачеві або конкретній новині.

```python
class News(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String, nullable=False)
    url = db.Column(db.String, unique=True, nullable=False)
    published = db.Column(db.DateTime, nullable=False)
    text = db.Column(db.Text, nullable=True)

    def comments_count(self):
        return Comment.query.filter(Comment.news_id == self.id).count()

    def __repr__(self):
        return 'News {} {}'.format(self.title, self.url)
```

Рис.39.Модель Новин з функціоналом підраховування коментарів

```python
@blueprint.route('/news/comment', methods=['POST'])
@login_required
def add_comment():
    form = CommentForm()
    if form.validate_on_submit():
            comment = Comment(text=form.comment_text.data, news_id=form.news_id.data, user_id=current_user.id)
            db.session.add(comment)
            db.session.commit()
            flash('Succefully added')
    else:
        for field, errors in form.errors.items():
            for error in errors:
                flash('Mistake in field {} {}'.format(
                    getattr(form, field).label.text,
                    error
                ))
    return redirect(get_redirect_target())
```

Рис.40.Функція коментарів

Функція add_comment() є маршрутом веб-додатка для додавання коментарів до новин. Функція перевіряє, чи форма коментарів валідна та додає коментар до бази даних. В іншому випадку вона відображає повідомлення про помилки на сторінку.

```python
class CommentForm(FlaskForm):
    news_id = HiddenField('News ID', validators=[DataRequired()])
    comment_text = StringField("Your comment:", validators=[DataRequired()], render_kw={"class": "form-control"})
    submit = SubmitField('Send comment', render_kw={"class": "btn btn-primary"})

    def validate_news_id(self, news_id):
        if not News.query.get(news_id.data):
            raise ValidationError('Article with this ID doesnt exist')
```

Рис.41. Рендер форми для коментарів

Клас CommentForm описує форму для додавання коментаря до новини. Він успадковує клас FlaskForm.

news_id - приховане поле з ID новини, до якої додається коментар.

comment_text - текстове поле для введення коментаря.

submit - кнопка для відправки форми.

Метод validate_news_id виконує перевірку наявності новини з вказаним ID в базі даних. Якщо новина не знайдена, викликається помилка валідації з повідомленням "Article with this ID doesnt exist"

```
<form action="{{ url_for('news.add_comment') }}" method="POST" class="comm">
    {{ comment_form.hidden_tag() }}
    <div class="form-group">
        {{ comment_form.comment_text.label }}
        {{ comment_form.comment_text() }}
    </div>
    <div class="form-group">
    {{ comment_form.submit() }}
    </div>
</form>
```

Рис.42. Застосування кментарів в HTML

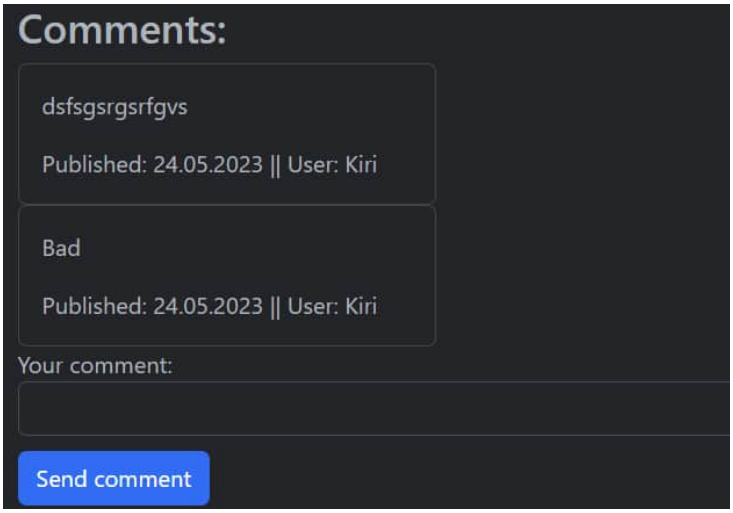


Рис.43. Інтерфейс коментарів в додатку

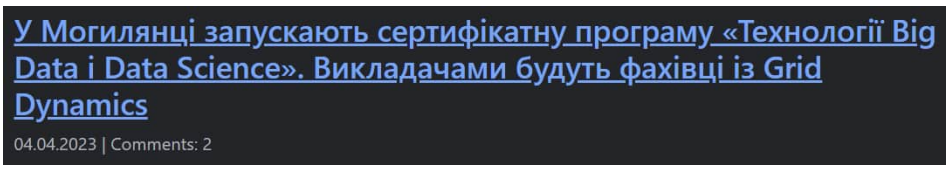


Рис.44. Блок новини з підрахованими коментарями

Висновки:

В результаті проведення дослідження та розробки проекту було отримано важливі висновки щодо використання мови програмування Python у створенні веб-додатків.

Виявлено, що Python є потужним і гнучким інструментом для розробки веб-додатків, завдяки своїй простоті, читабельності коду та багатому екосистемі бібліотек і фреймворків. Фреймворк Flask, який був досліджений у проекті, дозволяє швидко та ефективно розробляти бекенд веб-додатків з використанням Python.

Одним із ключових висновків є те, що Python може замінити JavaScript у веб-розробці, особливо в контексті створення бекенду. Використання фреймворка Flask та баз даних спрощує розробку веб-додатків і дозволяє зосередитися на логіці програми та взаємодії з іншими сервісами.

Крім того, досліджено можливості використання Python для роботи з базами даних, взаємодії з веб-сервісами та створення аналітичних інструментів. Виявлено, що Python має потужні бібліотеки та інструменти для роботи з різними типами баз даних та API, що робить його відмінним вибором для розробки веб-додатків з аналітичним функціоналом.

Загальним результатом проекту є розробка веб-додатку з використанням Python та Flask, який демонструє можливості мови програмування Python у веб-розробці. Розроблений додаток підтверджує, що Python є ефективним і зручним інструментом для створення веб-додатків з використанням простоти та читабельності коду.

Перспективи використання: Веб-проект можна використовувати як агрегатор новин з багатьох джерел, також веб-проект надає змогу залишати коментарі на записах, тобто якщо в джерелі новини не буде такої можливості, проект надасть потрібний функціонал.

Шляхи покращення: Даний веб-проект можна розвивати в багатьох напрямках, наприклад реалізувати можливість додавати джерела новин будь-якому користувачу. Також можна реалізувати фільтри новин, щоб користувач міг відсіяти непотрібні йому джерела новин або виводити новини лише по конкретним запитам.

Отже, дипломна робота успішно досягла своєї мети, розкривши повний потенціал мови програмування Python у веб-розробці та продемонструвавши можливість створення веб-додатків без необхідності використання JavaScript. Результати дослідження можуть бути корисними для розробників, які цікавляться використанням Python у веб-розробці та бажають зосередитися на простоті та ефективності розробки веб-додатків.

Використані джерела

1. Alembic: [Електронний ресурс]. -Режим доступу: https://alembic.sqlalchemy.org/en/latest/
2. Celery [Електронний ресурс]. -Режим доступу: https://docs.celeryq.dev/en/stable/
3. SQLAlchemy [Електронний ресурс]. -Режим доступу: https://docs.sqlalchemy.org/en/20/
4. VS Code [Електронний ресурс]. -Режим доступу: https://code.visualstudio.com/docs
5. DB Browser [Електронний ресурс]. -Режим доступу: https://sqlitebrowser.org/
6. Git [Електронний ресурс]. -Режим доступу: https://git-scm.com/doc
7. GitHub [Електронний ресурс]. -Режим доступу: https://docs.github.com/en
8. Python 3.11 [Електронний ресурс] . -Режим доступу: https://docs.python.org/3.11/
9. Flask [Електронний ресурс]. -Режим доступу: https://flask.palletsprojects.com/en/2.3.x/
10. WTForms [Електронний ресурс]. -Режим доступу: https://wtforms.readthedocs.io/en/3.0.x/
11. Bootstrap [Електронний ресурс]. -Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/
12. WSL [Електронний ресурс]. -Режим доступу: https://learn.microsoft.com/en-us/windows/wsl/
13. HTML [Електронний ресурс]. -Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML
14. CSS [Електронний ресурс]. -Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS
15. Javascript [Електронний ресурс]. -Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
16. Redis [Електронний ресурс]. -Режим доступу: https://redis.io/docs/
17. Werkzeug [Електронний ресурс]. -Режим доступу: https://werkzeug.palletsprojects.com/en/2.3.x/
18. BeautifulSoup4 [Електронний ресурс]. -Режим доступу: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
19. Flask-WTF [Електронний ресурс] – https://flask-wtf.readthedocs.io/en/1.0.x/
20. Flask-login [Електронний ресурс] – https://flask-login.readthedocs.io/en/latest/
21. Flask-SQLAlchemy [Електронний ресурс] – https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/
22. Flask-migrate [Електронний ресурс] – https://flask-migrate.readthedocs.io/en/latest/

Додаток А. Лістинг коду проекту

diploma/webapp

__init__.py

```
from flask import Flask, render_template

from flask_login import LoginManager

from flask_migrate import Migrate


from webapp.admin.views import blueprint as admin_blueprint

from webapp.news.views import blueprint as news_blueprint

from webapp.db import db

from webapp.user.models import User

from webapp.user.views import blueprint as user_blueprint


def create_app():

app = Flask(__name__)

app.config.from_pyfile('config.py')

db.init_app(app)

migrate = Migrate(app,db)


login_manger = LoginManager()

login_manger.init_app(app)

login_manger.login_view = 'user.login'

app.register_blueprint(admin_blueprint)

app.register_blueprint(user_blueprint)

app.register_blueprint(news_blueprint)


@login_manger.user_loader

def load_user(user_id):

return User.query.get(user_id)


return app
```

```
#Функція create_app() створює Flask додаток з конфігурацією з файлу config.py,

# підключує базу даних SQLAlchemy та міграції з Flask-Migrate.

# Далі, ініціалізує об'єкт LoginManager для авторизації користувачів та зареєстровує блюпрінти,

# які містять маршрути та логіку відповідної частини додатку. Також визначає функцію load_user(),

# яка повертає об'єкт користувача на основі ідентифікатора користувача.
```

diploma/

webapp/news/parsers/

dou.py

```
from datetime import datetime

from webapp.db import db


from webapp.news.models import News

from bs4 import BeautifulSoup

from webapp.news.parsers.utils import get_html, save_news


def get_news_snippets():

html = get_html("https://dou.ua/lenta/tags/python/?lang=en")

if html:

soup = BeautifulSoup(html, 'html.parser')

all_news = soup.find('div', class_='b-lenta').findAll('article', class_='b-postcard')

result_news = []

for news in all_news:
```

```
find_title = news.find('h2', class_='title').find('a')
title = find_title.text.strip()
url = news.find('h2', class_='title').find('a')['href']
published = news.find('time', class_='date').text

try:
published = datetime.strptime(published, '%d %B, %H:%M').replace(year=datetime.now().year)
except ValueError:
published = datetime.strptime(published, '%d %B %Y, %H:%M')
save_news(title, url, published)

def get_news_content():
news_without_text = News.query.filter(News.text.is_(None))
for news in news_without_text:
html = get_html(news.url)
if html:
soup = BeautifulSoup(html, 'html.parser')
news_text = soup.find('article', class_='b-typo b-typo_post').decode_contents()
if news_text:
news.text = news_text
db.session.add(news)
db.session.commit()
```

webapp/static

style.css

```
.news-content img {
max-width: 50%;
}
.tg-promo img{
max-width: 5%;
}

input[type="submit"] {
margin-top: 10px;
}
```

webapp/templates/

templates/user

registration.html

```
{% extends 'base.html' %}

{% block content %}
<h1><div class="mx-auto p-2" style="width: 300px;">{{ page_title }}</div></h1>
{% include('messages.html') %}
<div class="row" >
<div class="mx-auto p-2" style="width: 300px;">
<form action="{{ url_for('user.process_reg') }}" method="POST">
{{ form.hidden_tag() }}
<div class="mb-3">
{{ form.username.label }}
{{ form.username() }}
</div>
<div class="mb-3">
{{ form.email.label }}
{{ form.email() }}
</div>
<div class="mb-3">
{{ form.password.label }}
{{ form.password() }}
</div>
```

```
<div class="mb-3">
{{ form.password2.label }}
{{ form.password2() }}
</div>
{{ form.submit() }}
</form>
</div>
{% endblock %}
```

webapp/user

views.py

```
from webapp.user.forms import LoginForm, RegistrationForm
from webapp.user.models import User
from webapp.db import db

from flask import Blueprint, flash, redirect, url_for, render_template
from flask_login import current_user, login_user, logout_user
from webapp.utils import get_redirect_target

blueprint= Blueprint('user', __name__, url_prefix='/users')

@blueprint.route('/login')
def login():
if current_user.is_authenticated:
return redirect(get_redirect_target())
title = 'Authorization'
login_form = LoginForm()
return render_template('user/login.html', page_title = title, form = login_form)

@blueprint.route('/process-login', methods=['POST'])
def process_login():
form = LoginForm()

if form.validate_on_submit():
user = User.query.filter(User.username == form.username.data).first()
if user and user.check_password(form.password.data):
login_user(user, remember=form.remember_me.data)
flash("You are logged in")
return redirect(url_for('news.index'))

flash('Mistake in username or password')
return redirect(url_for('user.login'))

@blueprint.route('/logout')
def logout():
logout_user()
flash('Logged out')
return redirect(url_for('news.index'))

@blueprint.route('/register')
def register():
if current_user.is_authenticated:
return redirect(url_for('news.index'))
title = 'Registration'
form = RegistrationForm()
return render_template('user/registration.html', page_title = title, form = form)

@blueprint.route('/process-reg', methods=['POST'])
def process_reg():
form = RegistrationForm()
if form.validate_on_submit():
```

```
news_user = User(username=form.username.data, email=form.email.data, role='user')

news_user.set_password(form.password.data)

db.session.add(news_user)

db.session.commit()

flash('You are passed a registration')

return redirect(url_for('user.login'))

else:

for field, errors in form.errors.items():

for error in errors:

flash('Problem in {} : {}'.format(

getattr(form,field).label.text,

error

))

flash('Please correct the errors in the form')

return redirect(url_for('user.register'))
```

```
#код для Flask-блупрінту, який обробляє сторінки веб-додатку пов'язані з аутентифікацією та реєстрацією користувачів.

#Перші дві функції - це сторінки для входу користувача.

# Функція login показує сторінку входу, а функція process_login перевіряє, чи вірні введені ім'я користувача та пароль.

# Якщо вони вірні, вона залогінює користувача та перенаправить його на сторінку з новинами.

# В іншому випадку вона поверне його на сторінку входу з помилкою.

#Функція logout від'єднує користувача від системи та перенаправляє його на головну сторінку з повідомленням про вихід.

#Функції register та process_reg відповідають за реєстрацію користувача. Функція register показує сторінку реєстрації,

# а функція process_reg перевіряє правильність введених даних та додає нового користувача в базу даних,

# якщо вони вірні. Якщо ж дані не вірні,

# вона повертає користувача на сторінку реєстрації з відповідною помилкою.

#В цьому коді також використовується шаблонизатор Jinja2 для рендерингу HTML-сторінок.
```