Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут «Українська інженерно-педагогічна академія»
Кафедра Інформаційних комп`ютерних технологій і математики

КВАЛІФІКАЦІЙНА РОБОТА

магістра

на тему

Професійна підготовка фахівців з комп'ютерних технологій для створення веб-додатків на Python з використанням бази даних SQLite
(тема кваліфікаційної роботи)

Виконав: студент 2 курсу, групи ПОЦТ23 мг
спеціальності: 015 Професійна освіта (Цифрові технології)
(шифр і назва напряму підготовки, спеціальності)

_____/ Ігор БОНДАРЕНКО
(підпис) (ім'я та прізвище)
Керівник _____/ Олександр КУПРІЯНОВ
(підпис) (ім'я та прізвище)
Рецензент _____/ Сергій РОМАНОВ
(підпис) (ім'я та прізвище)

«До захисту допущено»

Завідувач кафедри ____/ Олеся НЕЧУЙВІТЕР
(підпис) (ім'я та прізвище)
Нормоконтроль_____/Сергій ТРОХІМЧУК
(підпис) (ім'я та прізвище)
Секретар ЕК _____/ Любов СОБОЛЬ
(підпис) (ім'я та прізвище)

Харків – 2024 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА

Навчально-науковий інститут «Українська інженерно-педагогічна академія»
Кафедра Інформаційних комп`ютерних технологій і математики
спеціальність 015 Професійна освіта (Комп'ютерні технології)
Освітньо-професійна програма Професійна освіта (Комп'ютерні технології)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____
д.ф.- м.н., доц., Олеся НЕЧУЙВІТЕР
«___» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу
Другого (магістерського) **рівня вищої освіти**

студенту (ці) *Бондаренко Ігорю Володимировичу*

1. Тема «Професійна підготовка фахівців з комп'ютерних технологій для створення веб-додатків на Python з використанням бази даних SQLite»
затверджена наказом по академії № 4801-5/3704 від "21" листопада 2024 р.
2. Термін здачі закінченої роботи "03" грудня 2024 р.
3. Вихідні дані до роботи а) приклади веб-додатків для управління документообігом; б) інструкції створення та налаштування веб-додатків; с) знання з методики професійного навчання.
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) актуальність професійної підготовки фахівців в галузі цифрових технологій, технології створення веб-додатків на Python та зберігання інформації в базі даних SQLite, вимоги до кадрового забезпечення фахівців зі створення інформаційних систем, методичний розділ.
5. Перелік графічного (ілюстративного) матеріалу код веб-додатку Newspaper Agency, презентація результатів кваліфікаційної роботи.
6. Консультант:

Розділ	Консультант	Підпис, дата		Оцінка (бали)
		Завдання видав	Завдання прийняв	
Методичний	Бачієва Лариса Олександрівна	04.11.2024	02.12.2024	

7. Дата видачі завдання "**04**" **жовтня 2024 р.**

Керівник роботи _____ **Олександр КУПРІЯНОВ**
(підпис)

Завдання прийняв до виконання _____ **Ігор БОНДАРЕНКО**
(підпис)

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

виконання дипломної роботи

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання	Позначки керівника про виконання завдань
1.	З’ясування поставленого завдання і оформлення завдання на дипломну роботу	05.10.2024	
2.	Огляд літератури стосовно розробки веб-сайтів	15.10.2024	
3.	Обґрунтування актуальності створення веб-сайтів	25.10.2024	
4.	Аналіз предметної області, опис застосування веб-сайтів	05.11.2024	
5.	Опис та налаштування середовища для створення веб-сайту	10.11.2024	
6.	Налаштування змісту веб-сайту	20.11.2024	
7.	Оформлення пояснювальної записки і графічного матеріалу	25.11.2024	
8.	Рецензування роботи	05.12.2024	
9.	Підготовка до захисту	10.12.2024	
10.	Захист дипломної роботи	16.12.2024	

Студент (ка) _____ Ігор БОНДАРЕНКО
(підпис)

Норм. контроль _____ Сергій ТРОХИМЧУК
(підпис)

РЕФЕРАТ

Робота містить: 80 с., 20 рис., 10 табл., 16 посилань.

Мета дослідження: визначення основних вимог до написання веб додатків, обґрунтування технологій, розробка програмного забезпечення, тестування та впровадження готового рішення, написання методичних вказівок до факультативного заняття.

Об’єкт дослідження: веб-додаток про управління газетним агентством, що включає інтерфейс користувача, базу даних та backend.

Предмет дослідження: процеси розробки та професійної підготовки фахівців з комп'ютерних технологій для створення веб-додатків на технології Python з використанням бази даних SQLite.

Методи дослідження: розробка програмного забезпечення, порівняльний аналіз, моделювання, тестування.

Завдання дослідження: розробити веб-додаток для управління газетним агентством, обґрунтувавши вибір технологій, реалізувавши процес створення та тестування програмного забезпечення, а також розробити методичні вказівки для факультативного заняття, використовуючи Python і SQLite.

Результати роботи дають можливість використання створеного програмного продукту в реальних умовах газетних агентств, а також застосуванні розроблених методичних вказівок для навчання студентів факультативним курсам з веб-програмування.

КЛЮЧОВІ СЛОВА: PYTHON, DJANGO, SQLITE, ВЕБ-ДОДАТОК, БАЗИ ДАНИХ, РОЗРОБКА.

ESSAY

Work contains: 80 pages, 20 figures, 10 tables, and 16 references..

Objective of the study: to define the main requirements for developing web applications, justify the selection of technologies, develop software, test and implement the final solution, and prepare methodological guidelines for an elective class.

Object of the study: a web application for managing a newspaper agency, including a user interface, database, and backend.
Subject of the study: the processes of development and professional training of IT specialists in creating web applications using Python technology with an SQLite database.

Research methods: software development, comparative analysis, modeling, and testing.

Research tasks: to develop a web application for managing a newspaper agency by substantiating the choice of technologies, implementing the development and testing processes, and creating methodological guidelines for an elective class using Python and SQLite.

Work results: The developed software product can be applied in real-world conditions of newspaper agencies, and the developed methodological guidelines can be used for teaching students elective courses in web programming.

KEYWORDS: PYTHON, DJANGO, SQLITE, WEB APPLICATION, DATABASE, DEVELOPMENT.

ЗМІСТ

ВСТУП

У сучасному світі, де інформаційні технології невпинно розвиваються, питання ефективної організації та обробки даних стає надзвичайно важливим, тому підготовка кадрів для напрямку науки про дані має як попит зі сторони абітурієнтів, так і пропозицію зі сторони університетів. Освітні заклади викладають широкий спектр технологій обробки даних, серед яких мова програмування Python доволі популярна. Причина, звичайно, в тому, що Python займає лідерські позиції в напрямку обробки даних.

SQL-запити майже повністю покладаються на комбінації простих методів до обробки даних. Це чудово підходить для нескладного аналізу, більш складні випадки використання потребують спеціальної мови програмування, де краще підходить Python. До Python створені значна кількість бібліотек і пакетів обробки даних, що дає значну перевагу у порівняння з обмеженнями функцій та операцій SQL.

В умовах інтеграції інформаційних технологій в освітні процеси, створення методичних матеріалів до навчання системам управління інформацією набуває значення. Розробка таких систем не лише забезпечує ефективність навчання, але й підвищує доступність інформації, що є важливим аспектом для покращення освітнього середовища.

Предметом дослідження є процес професійної підготовки фахівців з комп'ютерних технологій, які займаються розробкою веб-додатків на Python з використанням бази даних SQLite. *Об'єктом дослідження* виступає процес фахової підготовки таких фахівців, а також технології, що застосовуються при розробці програмних комплексів для управління інформацією.

Метою дипломної роботи є аналіз, теоретичне обґрунтування та вдосконалення процесу професійної підготовки фахівців для розробки веб-додатків, які використовують Python та SQLite для ефективної обробки та зберігання даних. Визначення основних вимог до таких фахівців та рекомендації щодо вдосконалення навчальних програм з урахуванням сучасних технологічних тенденцій є важливою частиною цієї роботи.

Завданнями роботи є:

1. Визначити актуальність професійної підготовки фахівців для створення веб-додатків із використанням Python та SQLite.

2. Розглянути ключові технології та інструменти для розробки веб-додатків, включаючи фреймворк Django, який використовує Python для створення потужних і масштабованих веб-рішень.

3. Створити додаток із застосуванням Python та SQLite, на прикладі якого можна базувати процес навчання означеним технологіям.

4. Описати вимоги до фахівців у галузі веб-розробки та специфічні навички, необхідні для роботи з Python та SQLite.

5. Розробити методичні рекомендації щодо підготовки фахівців для ефективної реалізації таких проектів в умовах сучасних освітніх стандартів.

Наукова новизна роботи полягає у вдосконаленні підходів до професійної підготовки фахівців, які здатні працювати з сучасними інструментами для створення веб-додатків і ефективного управління даними за допомогою Python та SQLite. Розроблені в рамках роботи методи та рекомендації можуть бути використані для створення навчальних курсів, які відповідають вимогам сучасного ринку праці.

Практичне значення цієї роботи полягає у створенні моделей навчання, які дозволяють майбутнім фахівцям з комп'ютерних технологій набувати навичок програмування для створення веб-додатків із високою продуктивністю, використовуючи інструменти Python і SQLite. Це дозволить значно покращити якість освіти в галузі IT, що відповідає потребам ринку та вимогам сучасних технологій.

Перспективи подальшого розвитку. Подальший розвиток дослідження може бути спрямований на вдосконалення функціональних можливостей створеного веб-додатка для управління газетним агентством, таких як інтеграція модуля аналітики для аналізу даних, автоматизація маркетингових кампаній та підтримка багатокористувацького доступу з диференціацією прав. Важливим кроком є застосування сучасних технологій, наприклад,

використання фронтенд-фреймворків React або Vue.js для покращення інтерфейсу, перехід на базу даних PostgreSQL для забезпечення масштабованості, а також інтеграція алгоритмів машинного навчання для персоналізації контенту. Особливу увагу варто приділити підвищенню продуктивності системи та її безпеки, зокрема оптимізації швидкості обробки запитів і захисту даних.

Крім того, перспективним є адаптування веб-додатка для інших сфер, таких як управління видавництвами або цифровими медіа. Значний потенціал має подальший розвиток освітнього аспекту дослідження, включаючи розробку спеціалізованих навчальних курсів, семінарів та публікацію методичних матеріалів для підготовки студентів у галузі веб-розробки. Комерціалізація результатів також відкриває можливості виходу на ринок із готовим програмним продуктом, його адаптації до потреб замовників і надання технічної підтримки. Загалом, дослідження має великий потенціал для розвитку як у науковій, так і практичній площинах, сприяючи цифровій трансформації медіа-галузі та вдосконаленню професійної підготовки фахівців у сфері комп'ютерних технологій.

Апробація роботи була здійснена шляхом публікації тез наукових доповідей, які висвітлювали ключові аспекти розробки та практичного впровадження системи, а також її значення у контексті професійної підготовки фахівців:

Бондаренко І.В. Професійна підготовка фахівців з комп’ютерних технологій для створення веб-додатків на Python з використання бази даних SQLITE. Збірник тез доповідей LIX студентської наукової конференції Української інженерно-педагогічної академії (м. Харків, 18-22 листопада 2024 р.) / Навчально-науковий інститут «Українська інженерно-педагогічна академія»; за заг. ред. Г.С. Грінченко.:, Харків, 2024. С. 85. (https://science.uipa.edu.ua/wp-content/uploads/2024/12/collection-of-abstracts.pdf)

1. АКТУАЛЬНІСТЬ ПРОФЕСІЙНОЇ ПІДГОТОВКИ ФАХІВЦІВ ДЛЯ СТВОРЕННЯ ВЕБ-ДОДАТКІВ НА PYTHON З ВИКОРИСТАННЯМ SQLITE

У сучасному світі інформаційних технологій веб-додатки стали невід'ємною частиною бізнесу, освіти, медицини, сфери розваг та інших галузей. Розвиток цифрової економіки сприяє зростанню попиту на спеціалістів, здатних створювати ефективні та безпечні веб-рішення. Однією з ключових мов програмування, яка забезпечує високу продуктивність у створенні веб-додатків, є Python. Завдяки своїй простоті, гнучкості та багатству бібліотек Python є одним із найбільш затребуваних інструментів серед розробників.

Окреме значення має використання баз даних, які забезпечують зберігання, обробку та доступ до інформації. У цьому контексті **SQLite** є одним із найпопулярніших рішень через свою простоту, невеликі системні вимоги та можливість інтеграції у веб-додатки будь-якого рівня складності. SQLite є ідеальним вибором для невеликих і середніх проєктів завдяки відсутності необхідності в окремому сервері та легкості використання.

Професійна підготовка фахівців у цій сфері потребує уваги до інтеграції сучасних технологій, що дозволяють забезпечити ефективність розробки та підтримки веб-додатків. Важливим аспектом є формування у студентів практичних навичок роботи з Python та SQLite, що сприяє підвищенню їхньої конкурентоспроможності на ринку праці.

Актуальність теми також зумовлена необхідністю забезпечення підприємств спеціалістами, які володіють сучасними технологіями програмування і баз даних для створення високоякісних веб-додатків. В умовах стрімкого зростання обсягів даних і запитів на їх обробку, фахівці мають володіти ефективними методами інтеграції інформаційних систем із застосуванням інструментів, що відповідають сучасним стандартам.

З огляду на це, вивчення методів створення веб-додатків на Python із використанням SQLite не лише забезпечує професійне зростання фахівців, але й сприяє розвитку економіки та суспільства в цілому. Вибір такої теми для

дослідження є надзвичайно актуальним, адже дозволяє вдосконалити процес професійної підготовки майбутніх програмістів та підвищити рівень їхньої готовності до вирішення практичних завдань у сфері веб-розробки.

1.1. Галузі і приклади застосування баз даних

Веб-додаток – це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера. Зовнішній інтерфейс веб програми розробляється за допомогою таких мов програмування: HTML, CSS, Javascript, які підтримуються на будь-якому браузері (Opera, Chrome, Mozilla). У той час як для написання серверної частини (Back-end) може використовуватися будь-яка інша мова програмування або фреймворк, Python, PhP, Ruby, Java.

Веб-додатки є невід'ємною частиною сучасної цифрової інфраструктури, забезпечуючи інтерактивність, доступність і функціональність для користувачів у різних галузях. Їх широке застосування зумовлене розвитком технологій інтернету, зростанням обсягів даних та потребою в автоматизації процесів.

Веб-продукти можна умовно поділити на три основні категорії, кожна з яких має свої особливості:

- **SPA (Single Page Application)** – односторінкові додатки, що забезпечують динамічне оновлення елементів. У таких додатках оновлюється лише потрібний користувачу контент, залишаючи основну сторінку статичною. Головною перевагою SPA є швидке завантаження завдяки мінімальному обміну даними із сервером.
- **MPA (Multi Page Application)** – багатосторінкові додатки, які взаємодіють із сервером постійно, перезавантажуючи сторінку після кожної дії. На відміну від SPA, MPA забезпечують кращу оптимізацію для пошукових систем і підходять для складних проектів, таких як платформи електронної комерції.
- **PWA (Progressive Web Application)** – прогресивні додатки, які поєднують властивості мобільних і веб-додатків. Їх можна використовувати як у браузері, так і встановлювати на пристрої у вигляді іконки. Ці додатки частково функціонують навіть без підключення до інтернету.

Крім класифікації за архітектурою, веб-додатки поділяють за призначенням:

- платформи для електронної комерції;
- соціальні мережі;
- освітні ресурси;
- системи для аналізу даних;
- корпоративні портали для внутрішніх процесів;
- сервіси для бронювання (готелів, квитків тощо);
- платформи для кур’єрської доставки.

Розробка веб-додатків та мобільних рішень надає компаніям можливість налагодити ефективну взаємодію з аудиторією, що позитивно впливає на продажі та розвиток бізнесу. Завдяки використанню сучасних технологій веб-додатки сприяють покращенню доступу до послуг, оптимізації процесів і підвищенню зручності для користувачів, що робить їх важливими інструментами для вирішення актуальних завдань у різних сферах діяльності.

1.2. Галузі та приклади використання баз даних SQLite

SQLite — це легковажна реляційна база даних, яка використовується у багатьох сферах завдяки своїй простоті, незалежності від серверів і високій продуктивності. Її застосування охоплює різноманітні галузі, де потрібна локальна обробка даних, зберігання невеликих обсягів інформації або швидкий доступ до бази.

Оскільки база даних SQLite не потребує адміністрування, вона добре працює на пристроях, які повинні працювати без експертної підтримки людини. SQLite добре підходить для використання в мобільних телефонах, приставках, телевізорах, ігрових приставках, камерах, годинниках, кухонній техніці, термостатах, автомобілях, верстатах, літаках, дистанційних датчиках, дронах, медичних пристроях і роботах: «Інтернет речей». Двигуни баз даних клієнт/сервер створені для роботи в центрі обробки даних, який є ядром мережі. SQLite також працює там, але SQLite також процвітає на межі мережі, дбаючи

про себе, забезпечуючи швидкі та надійні служби даних для програм, які інакше мали б хитромудрий зв’язок.

SQLite чудово працює як механізм баз даних для більшості веб-сайтів із низьким і середнім трафіком. Обсяг веб-трафіку, який може обробити SQLite, залежить від того, наскільки веб-сайт використовує свою базу даних. Люди, які розуміються на SQL, можуть використовувати оболонку командного рядка sqlite3 для аналізу великих наборів даних. Необроблені дані можна імпортувати з файлів CSV, а потім ці дані можна нарізати на шматочки для створення безлічі зведених звітів. Більш складний аналіз можна виконати за допомогою простих сценаріїв, написаних на Tcl або Python (обидва з них мають вбудований SQLite), або на R чи інших мовах за допомогою доступних адаптерів. Можливе використання включає аналіз журналу веб-сайту, аналіз спортивної статистики, компіляцію показників програмування та аналіз експериментальних результатів. Багато дослідників біоінформатики використовують SQLite таким чином.

Звичайно, те саме можна зробити з корпоративною базою даних клієнт/сервер. Перевага SQLite полягає в тому, що його легше встановити та використовувати, а отримана база даних є одним файлом, який можна записати на USB-накопичувач або надіслати електронною поштою колезі. Розробники систем повідомляють про успішне використання SQLite як сховища даних у серверних програмах, які працюють у центрі обробки даних, або, іншими словами, використання SQLite як базового механізму зберігання для сервера бази даних, що відповідає програмі. З цим шаблоном загальна система все ще є клієнт/сервер: клієнти надсилають запити на сервер і отримують відповіді через мережу. Але замість того, щоб надсилати загальний SQL і повертати необроблений вміст таблиці, клієнтські запити та відповіді сервера є високорівневими та залежать від програми. Сервер перетворює запити на кілька запитів SQL, збирає результати, виконує постобробку, фільтрацію та аналіз, а потім створює відповідь високого рівня, що містить лише важливу інформацію. Розробники повідомляють, що в цьому сценарії SQLite часто швидше, ніж

клієнт/серверна база даних SQL. Запити до бази даних серіалізуються сервером, тому паралельність не є проблемою. Паралельність також покращується завдяки «шардингу бази даних»: використання окремих файлів бази даних для різних субдоменів. Наприклад, сервер може мати окрему базу даних SQLite для кожного користувача, отже

Загалом SQLite вважається доволі простою базою даних і часто використовується новачками в сфері IT та студентами, оскільки вона не потребує попередньої конфігурації. Однак, попри це, вона є універсальним інструментом, який знаходить застосування в різних галузях завдяки своїй простоті, доступності та широким можливостям інтеграції. Його використання дозволяє розробникам створювати ефективні рішення, які відповідають сучасним вимогам до продуктивності та мобільності.

1.3. Роль Python у сучасній веб-розробці

Python є однією з найпопулярніших мов програмування у сучасній веб-розробці завдяки своїй простоті, універсальності та великій кількості бібліотек і фреймворків. Згідно зі статистикою, Python входить до топ-3 мов, які використовуються в різних сферах IT, включно з веб-розробкою, що свідчить про його важливість у створенні якісних і масштабованих рішень.

Python відомий своєю легкою у вивченні та читабельною синтаксисом, що робить його ідеальним вибором для новачків і професіоналів. Завдяки мінімалістичному підходу до написання коду, веб-розробники можуть зосередитися на вирішенні бізнес-завдань, а не на складнощах програмування.

Саме завдяки фреймворкам Python може бути застосований у різних сферах веб-розробки. Він легко інтегрується з JavaScript для забезпечення динамічного користувацького інтерфейсу. Простий синтаксис дозволяє швидко та ефективно писати команди CI/CD та автоматизувати розгортання, що одними з основних принципів DevOps.

Python підтримує роботу з багатьма базами даних, включно з реляційними (MySQL, PostgreSQL, SQLite) та нереляційними (MongoDB, Redis). Завдяки бібліотекам, як-от SQLAlchemy та Django ORM, розробники можуть швидко

працювати з даними без необхідності писати складні SQL-запити вручну. Поєднання з інструментами машинного навчання (TensorFlow, PyTorch) дозволяють розробникам для створювати веб-додатки з функціями штучного інтелекту.

Python має низку переваг у веб-розробці. По-перше, його швидкість розробки забезпечується великою кількістю готових бібліотек і фреймворків, які дозволяють швидко створювати прототипи додатків і впроваджувати нові функції. По-друге, Python відзначається масштабованістю: веб-додатки, розроблені на цій мові, легко адаптуються до зростаючих потреб бізнесу. Нарешті, висока безпека досягається завдяки використанню перевірених бібліотек, таких як Django, що зменшує ризик появи уразливостей у веб-додатках. Наприклад, соціальна мережа Instagram створена з використанням Django, фреймворка на Python, і активно використовує цю мову в інфраструктурі. У сфері електронної комерції платформи на кшталт Shopify застосовують Python для обробки даних і розробки API. Також Python відіграє важливу роль у сервісах потокового передавання: YouTube використовує його для управління інфраструктурою та створення аналітичних інструментів.

Таким чином, Python відіграє ключову роль у сучасній веб-розробці завдяки своїм функціональним можливостям, потужним інструментам та здатності вирішувати завдання будь-якої складності. Це робить його незамінним інструментом для розробників, які прагнуть створювати якісні та інноваційні веб-додатки.

1.4. Висновки до 1 розділу

У першому розділі було проаналізовано сучасні тенденції та перспективи застосування веб-додатків і баз даних, а також роль Python у їх розробці. На основі проведеного аналізу можна зробити такі висновки:

1. Актуальність веб-додатків**:** Веб-додатки стали невід'ємною частиною сучасного цифрового світу, знаходячи застосування в бізнесі, освіті, медицині, фінансах, розвагах та інших галузях. Їх використання забезпечує автоматизацію

процесів, підвищення ефективності роботи організацій та покращення користувацького досвіду.

2. Значущість баз даних SQLite: Бази даних SQLite відіграють важливу роль у багатьох галузях завдяки їх простоті, легкій інтеграції та високій продуктивності. Їх застосування варіюється від мобільних додатків і вбудованих систем до аналітичних інструментів та автономних рішень.

3. Роль Python у веб-розробці: Python став однією з провідних мов програмування для розробки веб-додатків. Завдяки простоті синтаксису, наявності потужних фреймворків (Django, Flask, FastAPI) та широкій інтеграції з базами даних, Python дозволяє створювати як прості, так і масштабовані додатки, які відповідають сучасним вимогам.

4. Перспективи для фахівців: Розвиток цифрових технологій підкреслює необхідність підготовки висококваліфікованих спеціалістів, здатних працювати з Python, базами даних SQLite і сучасними фреймворками. Знання цих технологій відкриває значні можливості для працевлаштування та професійного зростання у сфері IT.

Отже, веб-додатки, бази даних SQLite та Python є ключовими компонентами сучасної цифрової екосистеми. Їхнє використання забезпечує розробку ефективних, безпечних та інноваційних рішень для різних галузей економіки та суспільства, а професійна підготовка фахівців у цій сфері є стратегічно важливим напрямком розвитку освіти та науки.

2. ТЕХНОЛОГІЇ РОЗРОБКИ ВЕБ-ДОДАТКІВ ТА ЗБЕРІГАННЯ ІНФОРМАЦІЇ В БАЗІ ДАНИХ SQLITE

Другий розділ дипломної роботи присвячений аналізу технологій розробки веб-додатків та зберігання інформації в базі даних SQLite за допомогою використання популярних фреймворків Python. Цей розділ надає ключову інформацію про роботу бази даних у синергії з вибраною мовою програмування.

2.1. Інструменти і бібліотеки для розробки веб-додатків

Python-фреймворки — це набір інструментів, бібліотек та модулів, які значно полегшують процес створення веб-додатків. Вони дозволяють розробникам зосередитися на логіці проєкту, оскільки вже містять готові рішення для типових завдань, таких як маршрутизація URL, взаємодія з базами даних чи валідація даних. Завдяки цьому створення веб-додатків стає ефективнішим, а код більш структурованим.

Основні можливості Python-фреймворків

Більшість популярних фреймворків включають:

- **Шаблони маршрутизації** для визначення обробників URL-адрес.
- **Інструменти валідації форм** для перевірки введених даних.
- **Механізми ORM** (об'єктно-реляційного відображення), що спрощують роботу з базами даних.
- **Захист від уразливостей**, таких як SQL-ін’єкції або підробка міжсайтових запитів (CSRF).
- **Інструменти для аутентифікації та авторизації користувачів**.
- **Системи шаблонізації** для формування HTML-сторінок.
- **Міграції баз даних**, що дозволяють оновлювати їх структуру без втрати даних.

Фреймворки також забезпечують незалежність від конкретної бази даних, дозволяючи використовувати один і той самий код із різними СУБД.

Класифікація Python-фреймворків

Python-фреймворки поділяються на три основні типи:

1. Фулстек-фреймворки — комплексні рішення для розробки повного циклу веб-додатків.

2. Мікрофреймворки — легкі платформи, які містять лише базовий функціонал, потребуючи додаткових налаштувань.

3. Асинхронні фреймворки — оптимізовані для роботи з великою кількістю одночасних з'єднань і реального часу.

Популярні Python-фреймворки

- **Django** — фулстек-фреймворк, який забезпечує всі необхідні інструменти для створення великих проєктів. Він пропонує ORM, шаблонізатор, систему аутентифікації, кешування та адміністрування. Django ідеально підходить для масштабних інтерактивних сайтів, що вимагають високої продуктивності.
- **Flask** — мікрофреймворк із мінімалістичним підходом. Надає лише основні інструменти, дозволяючи розробникам вибирати додаткові модулі залежно від потреб проєкту. Flask часто використовується для створення невеликих веб-додатків та API.
- **Pyramid** — універсальний фреймворк із модульною архітектурою, що підходить для створення як невеликих проєктів, так і масштабованих систем.
- **Tornado** — асинхронний фреймворк для високонавантажених додатків, здатний одночасно обробляти тисячі підключень завдяки підтримці вебсокетів та інших сучасних технологій.
- **Web2py** — фулстек-фреймворк, що забезпечує простоту розгортання, автоматичну генерацію SQL-запитів і зручність у використанні.
- **CherryPy** — мікрофреймворк із вбудованим високошвидкісним сервером WSGI, який дозволяє створювати швидкі та продуктивні веб-додатки.

Переваги використання Python-фреймворків

Фреймворки спрощують процес розробки, зменшують кількість повторюваних дій і допомагають дотримуватися кращих практик. Вибір відповідного фреймворку залежить від типу проєкту, його складності та вимог, що ставляться до продуктивності, масштабованості й безпеки.

2. 2. Особливості роботи з базами даних SQLite

SQLite — це вбудована реляційна база даних з відкритим кодом, яка вирізняється компактністю та широкою популярністю. Її використовують у численних застосунках, де потрібно зберігати дані локально. Завдяки своїй легкості та функціональності SQLite отримала нагороду Google-O'Reilly Open Source Awards і активно застосовується в різних сферах.

Основні характеристики SQLite

1. Вбудованість. SQLite не потребує серверного налаштування, оскільки це бібліотека, яку інтегрують безпосередньо у додаток. Таким чином, користувачеві не потрібно встановлювати чи адмініструвати сервер бази даних.

2. Мінімальні ресурси. База даних має низькі системні вимоги, що робить її ідеальною для середовищ з обмеженими ресурсами, наприклад, мобільних додатків.

3. Стандарт SQL. SQLite підтримує більшість можливостей SQL, що дозволяє ефективно керувати даними.

4. Простота інтеграції. Використовуючи API бібліотеки, програмісти можуть взаємодіяти з базою даних у різних мовах програмування без зайвих складнощів.

5. Транзакції. Підтримка транзакцій забезпечує цілісність даних навіть у багатозадачних процесах.

SQLite є оптимальним вибором для проектів, де немає потреби в масштабних серверних рішеннях. Її простота й ефективність роблять її популярною серед розробників мобільних, десктопних додатків та веббраузерів.

Архітектура SQLite

1. Файл бази даних. Всі дані SQLite зберігаються у файлі з розширенням `.sqlite` або `.db`. Цей файл можна легко копіювати, переміщати чи створювати резервні копії.

2. Таблиці та стовпці. Таблиці визначаються схемою, яка містить імена стовпців, типи даних і правила. Наприклад:

```
CREATE TABLE Users (
    ID INTEGER PRIMARY KEY,
    Name TEXT NOT NULL,
    Age INTEGER
);
```

У цьому прикладі створюється таблиця `Users` з полями `ID`, `Name` та `Age`.

Записи. Таблиці зберігають окремі рядки даних (записи). Наприклад:
INSERT INTO Users (ID, Name, Age) VALUES (1, 'John', 30);

Індекси. Індекси прискорюють пошук і доступ до даних. Їх створюють на стовпцях, які часто використовуються у запитах.

Транзакції. Транзакції об’єднують групу операцій у єдине ціле, забезпечуючи їх виконання без помилок або їх відкат у разі збою.

Переваги та недоліки SQLite

Переваги:

- Легка інтеграція без необхідності встановлення сервера.
- Простота у використанні та налаштуванні.
- Усі дані зберігаються в одному файлі, що спрощує резервне копіювання та перенесення.
- Підтримка транзакцій для збереження цілісності даних.
- Підходить для багатьох мов програмування (Python, Java, C# тощо).

Недоліки:

- Погано працює з великим обсягом одночасних записів (можливі блокування).

- Обмежена продуктивність для великих обсягів даних.
- Не підтримує віддалений доступ через мережу.
- Не підходить для складних аналітичних завдань.

Варіанти використання SQLite

SQLite використовується в різноманітних проєктах завдяки своїй універсальності:

- **Мобільні додатки:** зберігання локальних даних, налаштувань і журналів.
- **Десктопні програми:** створення локальних баз для невеликих застосунків.
- **Вбудовані системи:** роутери, GPS-пристрої, телевізори тощо.
- **Веббраузери:** зберігання історії, закладок (наприклад, у Google Chrome).
- **Прототипування:** швидке створення та тестування додатків.
- **Локальне кешування:** прискорення роботи додатків за рахунок тимчасового зберігання даних.

SQLite ідеально підходить для проєктів, які не вимагають масштабованості чи роботи з великими обсягами даних. Саме тому цій роботі було використано саме SQLite.

2.3. Процес інтеграції Python і SQLite у веб-додатках

У сучасному цифровому середовищі створення веб-додатків із функцією збереження та обробки даних є ключовим аспектом, що дозволяє організовувати, управляти та обмінюватися великими обсягами інформації. Веб-додаток — це програмний продукт, який забезпечує доступ до даних і сервісів через Інтернет, локальну корпоративну мережу або інші комунікаційні платформи. Такі рішення можуть застосовуватись для зберігання даних, їхньої обробки та інтеграції з іншими сервісами.

Python і SQLite, будучи потужними та легкими у використанні інструментами, можуть бути інтегровані для створення функціональних і

продуктивних веб-додатків. У цьому розділі акцентується увага на виборі технологій для **front-end** і **back-end**, їхній взаємодії та оптимізації процесів розробки.

2.3.1. Вибір програмного забезпечення для back-end частини

У даній дипломній роботі для розробки back-end частини буде використовуватися Python, а саме один із його найпопулярніших фреймворків - Django.

Django — це один із найпотужніших і найпопулярніших фулстек-фреймворків для розробки веб-додатків на Python. Його ключова перевага полягає у можливості швидкого створення проєктів будь-якої складності завдяки великій кількості вбудованих інструментів.

Основні особливості Django

Об'єктно-реляційне відображення (ORM): автоматизує роботу з базами даних, дозволяючи працювати з даними через Python-код без написання SQL-запитів.

Система шаблонів: спрощує створення динамічних HTML-сторінок із можливістю гнучкого налаштування.

Інструменти аутентифікації та авторизації: забезпечують безпеку користувацьких даних.

Адміністративна панель: дозволяє керувати контентом і даними без додаткового програмування.

Підтримка міграцій: спрощує процес оновлення структури баз даних.

Django часто використовується для розробки великих інтерактивних проєктів, які вимагають масштабованості, високої продуктивності та безпеки. Наприклад, соціальна мережа Instagram побудована на основі цього фреймворку.

Django автоматично підтримує SQLite як стандартну базу даних для невеликих або середніх проєктів. Це дозволяє швидко налаштовувати та запускати проєкти без необхідності попередньої конфігурації серверної інфраструктури. SQLite ідеально підходить для прототипів або невеликих систем, що мають обмежений обсяг даних.

2.3.2. Вибір програмного забезпечення для front-end частини

Розробка фронтенд-частини веб-додатків є важливим етапом створення сучасних інтерфейсів користувача, які забезпечують зручність взаємодії з додатком. Для реалізації front-end частини цього проекту будуть використані HTML та CSS — основні технології, які дозволяють створювати структуровані, адаптивні та візуально привабливі веб-сторінки.

HTML (HyperText Markup Language) забезпечує основу для створення веб-сторінки, визначаючи її структуру та вміст. Ця мова розмітки дозволяє додавати заголовки, текст, зображення, таблиці, форми та інші елементи, необхідні для функціонування додатку.

CSS (Cascading Style Sheets) використовується для оформлення елементів HTML. Це дає змогу створювати стильний і сучасний дизайн, що відповідає вимогам користувачів та бізнесу. CSS дозволяє налаштовувати кольори, шрифти, розміри, відступи, анімації та інші аспекти візуального вигляду веб-додатку.

2.4. Контроль якості веб-додатків

Контроль якості в Python є фундаментальним аспектом розробки програмного забезпечення, який відіграє вирішальну роль у забезпеченні надійності, правильності та зручності обслуговування вашого коду. Застосовуючи ефективні стратегії тестування, використовуючи надійні інфраструктури тестування та дотримуючись найкращих практик, можливо створювати високоякісні програми Python, які відповідають очікуванням користувачів і витримують виклики реального використання.

Тестування зазвичай поділяють на ручне та автоматизоване:

1. Ручне тестування – це тип тестування, у якому ми не використовуємо жодних інструментів (автоматизації) для виконання тестування. Під час цього тестування тестувальники створюють тестові випадки для кодів, перевіряють програмне забезпечення та дають остаточний звіт про це програмне забезпечення. Тестування вручну займає багато часу, оскільки це роблять люди, і існує ймовірність людських помилок.

2. Автоматизоване тестування — це тип тестування, у якому ми використовуємо інструменти (автоматизацію) для виконання тестування. Це швидше, ніж ручне тестування, оскільки воно виконується за допомогою деяких засобів автоматизації. Немає ймовірності людських помилок.

Автоматизоване тестування перевершує ручне завдяки швидкості, ефективності та точності. Воно дозволяє виконувати великі обсяги тестів за короткий час, повторювати їх багаторазово без зниження якості та уникати людських помилок. Інструменти автоматизації забезпечують стабільність результатів і дозволяють швидко виявляти помилки навіть у складних системах. Хоча початкові витрати на створення тестів можуть бути високими, у довгостроковій перспективі автоматизація економить ресурси та прискорює випуск якісного продукту.

2.5. Висновки до 2-го розділу

У другому розділі розглянуто ключові аспекти розробки веб-додатків із використанням Python та SQLite, що є важливими для створення сучасних і функціональних програмних продуктів. Детально проаналізовано інструменти та фреймворки, які значно спрощують процес розробки, забезпечуючи ефективність і структурованість коду.

Також акцентовано увагу на особливостях інтеграції SQLite як бази даних, що забезпечує швидке та легке управління даними завдяки її вбудованій архітектурі, простоті у використанні та низьким системним вимогам. SQLite обрана через її зручність для проєктів невеликого та середнього масштабу.

Окрім того, розглянуто процеси вибору технологій для front-end та back-end частин, що дозволяють створити зручний і привабливий інтерфейс користувача та забезпечити надійність роботи серверної частини.

Нарешті, підкреслено важливість контролю якості веб-додатків, де автоматизоване тестування визначено як ефективний метод забезпечення стабільності та функціональності програмного забезпечення. Це підкреслює

прагнення до створення якісних продуктів, що відповідають потребам користувачів та бізнесу.

3. ПРИКЛАД СТВОРЕННЯ ВЕБ-ДОДАТКУ НА PYTHON З ВИКОРИСТАННЯМ SQLITE

У попередніх розділах дипломної роботи ми зосередилися на теоретичних аспектах веб-розробки та створення інформаційних систем. Ці знання є основою для розуміння процесів, що відбуваються при розробці сучасних веб-додатків, і є критично важливими для вирішення складних завдань у галузі IT, зокрема в освітньому секторі. Тепер ми можемо перейти від теорії до практики, і в цьому розділі продемонструємо створення веб-додатку на Python із використанням бази даних SQLite.

Метою цього розділу є показати повний процес розробки веб-додатку, який використовує базу даних для зберігання інформації, зокрема для управління даними абітурієнтів у навчальному закладі. Ми зосередимося на виборі технологічного стеку, а також на ключових аспектах розробки, таких як забезпечення ефективності, безпеки та зручності використання системи.

Процес створення цього веб-додатку включатиме розробку моделей даних, реалізацію взаємодії з базою даних через SQLite, а також розробку інтерфейсу користувача, який буде зручним і інтуїтивно зрозумілим для адміністраторів навчального закладу. Одним із основних завдань буде створення системи, яка дозволить ефективно зберігати та обробляти дані абітурієнтів, а також забезпечити доступ до цієї інформації через веб-інтерфейс.

Вибір технологій, зокрема Python і SQLite, є обґрунтованим через їхню популярність та ефективність для розробки невеликих та середніх веб-додатків, що працюють з базами даних. База даних SQLite забезпечує простоту інтеграції і високу продуктивність при роботі з невеликими обсягами даних, що є характерним для освітніх установ. Крім того, фреймворк Django, який ми обрали для реалізації цього проєкту, дозволяє швидко і зручно розробляти веб-додатки, забезпечуючи необхідний рівень безпеки та зручності в роботі.

Цей розділ також включатиме розгляд основних проблем, з якими може зіштовхнутися розробник під час реалізації такої системи, і надасть рішення для забезпечення безпеки даних та зручності їхнього доступу. Ми також

обговоримо основні методи тестування та валідації даних, щоб забезпечити правильність роботи системи.

Отже, цей розділ не лише описує практичний аспект створення веб-додатку для управління інформацією, але й демонструє, як теоретичні знання з веб-розробки та баз даних можуть бути застосовані для розв'язання реальних задач у сучасному освітньому процесі.

3.1. Архітектура веб-додатку

У сучасній веб-розробці створення якісного веб-додатку виходить за межі написання коду. Важливою складовою професійної підготовки фахівців з комп’ютерних технологій є розуміння архітектури веб-додатків та принципів їхньої побудови. Архітектура веб-додатку не лише визначає, як організувати окремі компоненти, але й закладає основи для масштабованості, підтримуваності та ефективності системи. У цьому розділі розглянемо загальні принципи архітектури веб-додатків на основі фреймворка Django з використанням мови програмування Python та бази даних SQLite.

Веб-додаток можна уявити як складну систему, що складається з функціональних модулів, потоків даних та взаємодій із користувачем. Для забезпечення ефективності важливо застосовувати структурований підхід до проєктування. Django як один із найбільш популярних фреймворків Python надає потужні інструменти для побудови архітектури веб-додатків, дотримуючись принципів модульності, масштабованості та повторного використання компонентів.

Архітектура додатку в Django базується на патерні Model-View-Controller (MVC), який у Django реалізовано як Model-View-Template (MVT). Ця модель розподіляє відповідальність між трьома основними компонентами:

Model (Модель): Представляє структуру даних у вигляді об'єктно-реляційної моделі (ORM). Забезпечує взаємодію з базою даних, включаючи збереження, оновлення та отримання даних.

View (Уявлення): Визначає логіку обробки запитів і формування відповідей. Уявлення є посередником між користувачем та даними.

Template (Шаблон): Відповідає за відображення інтерфейсу користувача. Шаблони використовуються для динамічного формування HTML-сторінок. Розподіл завдань між цими компонентами сприяє підтримуваності коду, полегшує співпрацю розробників та дозволяє легко розширювати функціональність.

Фреймворк Django забезпечує стандартизовану структуру проєкту, яка полегшує розробку і організацію коду. При створенні нового проєкту за допомогою команди django-admin startproject генерується базова структура:

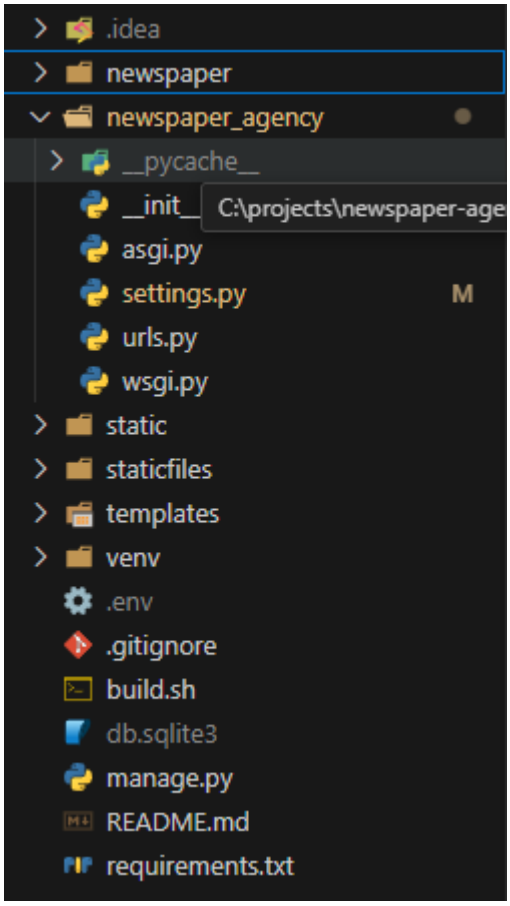

Рис. 3.1. Базова структура Django проєкту

Основні файли та їх функції:

1. manage.py – скрипт для управління проєктом: запуск сервера розробки, виконання міграцій, створення застосунків тощо.

2. settings.py – файл конфігурації, де визначаються налаштування проєкту, включаючи параметри бази даних, налаштування середовища тощо.

3. urls.py – файл маршрутизації, що відповідає за перенаправлення запитів до відповідних уявлень.

4. wsgi.py та asgi.py – файли, які забезпечують зв’язок додатку з вебсервером (у синхронному чи асинхронному режимах).

5. init.py – перетворює директорію проєкту в Python-пакет.

Кожен проєкт у Django складається із застосунків (apps), що є незалежними модулями, які виконують окремі функції, наприклад автентифікацію користувачів чи керування контентом. Застосунки створюються за допомогою команди: python manage.py startapp my_app.

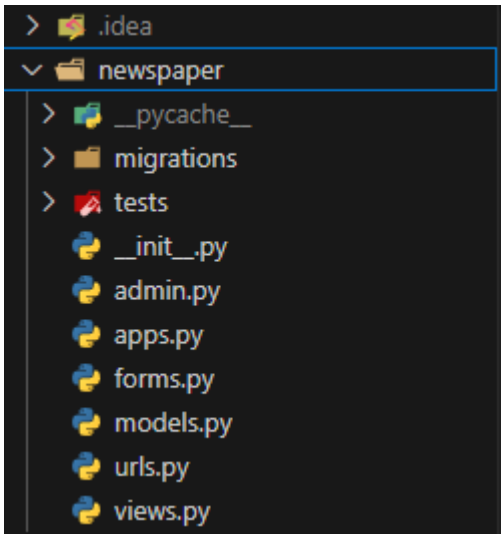

Рис. 3.2. Структура типової директорії застосунку

1. models.py – визначає моделі даних, що відповідають таблицям у базі даних.

2. views.py – містить логіку обробки запитів.

3. admin.py – налаштовує відображення моделей у панелі адміністратора Django.

4. urls.py – маршрутизація запитів у межах застосунку.

5. migrations/ – містить скрипти для синхронізації змін у моделях із базою даних.

Django підтримує принципи модульності та багаторазового використання компонентів. Застосунки в проєкті є самостійними модулями, що дозволяє їх перевикористовувати в інших проєктах, полегшує тестування та зменшує залежності між компонентами.

Архітектура веб-додатків у Django поєднує структурований підхід із гнучкістю та потужними інструментами розробки. Завдяки стандартизованій структурі проєкту, підтримці MVT-патерну та використанню інтегрованих рішень, таких як SQLite, розробка стає більш організованою та ефективною. Таке середовище сприяє підготовці кваліфікованих фахівців, здатних створювати масштабовані, підтримувані та продуктивні веб-додатки.

3.2. Програмування back-end частини

У сучасній розробці веб-додатків особливу увагу приділяють програмуванню backend частини, яка забезпечує основну логіку системи та взаємодію з базою даних. Для розробки backend частини веб-додатків на Python часто використовують фреймворк Django, який забезпечує зручне і швидке створення веб-додатків, зокрема завдяки інтеграції з базами даних, такими як SQLite.

Django надає потужні інструменти для організації моделей даних, що є основою backend частини додатка. Моделі описують структуру даних, з якими працює додаток, і відповідають за збереження цих даних у базі. Веб-додаток для управління інформацією про абітурієнтів, що розробляється за допомогою Django, може використовувати SQLite як реляційну базу даних, що є зручною для невеликих та середніх за розміром додатків завдяки своїй легкості та простоті в налаштуванні.

Backend частина веб-додатку забезпечує логіку взаємодії з користувачем через API або форми, що дозволяє виконувати різноманітні операції, такі як додавання, редагування, видалення інформації про абітурієнтів. Крім того, Django надає зручні інструменти для аутентифікації та авторизації користувачів, що забезпечує контроль доступу до даних.

Одним із основних завдань програмування backend частини є забезпечення безпеки даних. Для цього Django містить вбудовані механізми захисту від основних загроз, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS), а також автоматичне шифрування паролів користувачів. Крім того,

важливою складовою є оптимізація запитів до бази даних та забезпечення стабільності роботи додатку при високих навантаженнях.

Отже, для ефективної розробки backend частини веб-додатку на Python з використанням Django, фахівець повинен мати ґрунтовні знання про роботу з базами даних, організацію моделей даних, а також впровадження заходів безпеки для забезпечення захисту інформації.

models.py × Welcome

newspaper > models.py > Newspaper

```python
from django.conf import settings
from django.contrib.auth.models import AbstractUser
from django.db import models


class Redactor(AbstractUser):
    years_of_experience = models.PositiveIntegerField(default=0)

    def __str__(self):
        return f"{self.username} ({self.first_name} {self.last_name})"


class Topic(models.Model):
    name = models.CharField(max_length=63)

    def __str__(self):
        return self.name


class Newspaper(models.Model):
    title = models.CharField(max_length=255)
    content = models.TextField()
    published_date = models.DateField(auto_now_add=True)
    topics = models.ManyToManyField(Topic, related_name="newspapers")
    publishers = models.ManyToManyField(
        settings.AUTH_USER_MODEL, related_name="newspapers"
    )

    def __str__(self):
        return self.title

    class Meta:
        ordering = ["-published_date"]
```

Рис. 3.3. Модуль models.py

Модуль models.py є частиною моделі Django для керування контентом, що стосується редагування новинних матеріалів і взаємодії з користувачами.

Клас Redactor спадкує від стандартної моделі користувача Django AbstractUser, що означає, що цей клас зберігає стандартні властивості

користувача (ім'я користувача, email, пароль тощо), а також додаткове поле для зберігання досвіду редагування (years_of_experience).

Поле years_of_experience є числовим та зберігає кількість років досвіду роботи редактора, причому за замовчуванням воно дорівнює 0.

Методу __str__ надається представлення об'єкта у вигляді рядка з імені користувача та його повного імені (першого та прізвища). Це корисно для відображення інформації про редактора в адміністративній панелі Django.

Клас Topic описує модель теми (категорії) для газетних статей. Поле name зберігає назву теми, яке обмежене 63 символами. Метод __str__ надає зручне представлення об'єкта теми у вигляді її назви.

Клас Newspaper описує модель газетної статті.

Поле title зберігає заголовок статті.

Поле content містить основний текст статті.

Поле published_date містить дату публікації статті, автоматично встановлюється під час додавання нової статті.

Поле topics встановлює багато до багатьох зв'язків з темами. Це означає, що одна стаття може бути віднесена до кількох тем.

Поле publishers встановлює багато до багатьох зв'язків з користувачами (редакторами, які опублікували статтю). Використовує settings.AUTH_USER_MODEL, що дозволяє використовувати власну модель користувача.

Метод __str__ надає представлення об'єкта газети у вигляді її заголовка.

Мета-клас Meta визначає порядок сортування статей за датою публікації (новіші статті будуть першими).

Основні функції та команди:

- Моделі для користувачів та статей: Моделі користувачів (Redactor) та статей (Newspaper) зберігають та обробляють основні дані про редакторів та статті.
- Зв'язки багато до багатьох: За допомогою ManyToManyField створюються зв'язки між статтями і темами, а також між статтями і редакторами. Це

дозволяє одному редактору публікувати багато статей, а статтям можна надавати кілька тем.

- Поля за замовчуванням та автоматичне заповнення: Використання default для years_of_experience і auto_now_add для published_date дає змогу автоматично заповнювати ці поля без додаткового вводу користувачем.
- Створення адміністративного інтерфейсу: Завдяки методам __str__, ці моделі можуть бути зручнішими для відображення в адміністративній панелі Django (чи будь-якому іншому інтерфейсі, де відображаються дані).

Модуль views.py визначає представлення (views) для різних частин веб-сайту, пов'язаних з управлінням редакторами, темами та газетними статтями. Вони реалізують основні функціональні можливості для відображення, створення, оновлення та видалення записів у базі даних через Django класові представлення (class-based views, CBVs).

Функція `index`**:**

- Ця функція є обробником для головної сторінки сайту. Вона виконує кілька важливих завдань:
- Підраховує кількість редакторів, тем і газет у базі даних, використовуючи запити до моделей.
- Відстежує кількість відвідувань сторінки через механізм сесій, зберігаючи та оновлюючи лічильник.
- Визначає значення для контексту, яке потім передається в шаблон для відображення статистики на головній сторінці.

```python
def index(request):
    """View function for the home page of the site."""

    num_redactors = Redactor.objects.count()
    num_topics = Topic.objects.count()
    num_newspapers = Newspaper.objects.count()

    num_visits = request.session.get("num_visits", 0)
    request.session["num_visits"] = num_visits + 1

    context = {
        "num_redactors": num_redactors,
        "num_topics": num_topics,
        "num_newspapers": num_newspapers,
        "num_visits": num_visits + 1,
    }

    return render(request, "newspaper/index.html", context=context)
```

Рис. 3.4. Функція index

Клас `RedactorListView` **(Список редакторів):**

- Цей клас використовує `ListView` для відображення переліку всіх редакторів.
- Включає **пагінацію**: на сторінці буде показано лише 5 редакторів. Це допомагає зменшити навантаження на браузер при відображенні великої кількості записів.
- Додається **форма пошуку**, яка дозволяє фільтрувати редакторів за іменем користувача. Цей механізм працює через параметри запиту (`GET`), що дозволяє знаходити відповідних користувачів за їхнім ім'ям.

```python
class RedactorListView(ListView):
    model = Redactor
    paginate_by = 5

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(RedactorListView, self).get_context_data(**kwargs)

        username = self.request.GET.get("username", "")

        context["search_form"] = RedactorSearchForm(initial={"username": username})
        return context

    def get_queryset(self):
        form = RedactorSearchForm(self.request.GET)
        queryset = super().get_queryset()

        if form.is_valid():
            username = form.cleaned_data.get("username")
            if username:
                queryset = queryset.filter(username__icontains=username)
        return queryset
```

Рис. 3.5. Клас RedactorListView

Клас `RedactorDetailView` **(Детальний вид редактора):**

- Використовує `DetailView` для показу повної інформації про конкретного редактора.
- Повертає детальний перегляд обраного редактора, використовуючи відповідний шаблон для відображення його даних.

Клас `RedactorCreateView` **(Створення редактора):**

- Цей клас базується на `CreateView`, що дозволяє створювати нові записи про редакторів.
- Важливою особливістю є те, що створення редактора доступне лише авторизованим користувачам завдяки міксину `LoginRequiredMixin`.
- Після успішного додавання нового редактора, користувач буде перенаправлений на головну сторінку.

```python
class RedactorDetailView(DetailView):
    model = Redactor


class RedactorCreateView(LoginRequiredMixin, CreateView):
    model = Redactor
    form_class = RedactorCreationForm
    success_url = reverse_lazy("newspaper:index")
```

Рис. 3.6. RedactorDetailView та RedactorCreateView

Клас `RedactorUpdateView` **(Оновлення редактора):**

- Використовує `UpdateView` для редагування існуючого редактора.
- Після редагування інформації про редактора користувач автоматично буде направлений на список всіх редакторів.

Клас `RedactorDeleteView` **(Видалення редактора):**

- Базується на `DeleteView` і дозволяє видалити редактора з бази даних.
- Після успішного видалення редактора, користувача перенаправляє на сторінку зі списком редакторів.

```python
class RedactorUpdateView(LoginRequiredMixin, UpdateView):
    model = Redactor
    form_class = RedactorUpdateForm
    success_url = reverse_lazy("newspaper:redactors")


class RedactorDeleteView(LoginRequiredMixin, DeleteView):
    model = Redactor
    success_url = reverse_lazy("newspaper:redactors")
```

Рис. 3.7. RedactorUpdateView та RedactorDeleteView

Клас `TopicListView` **(Список тем):**

- Використовує `ListView` для виведення всіх доступних тем.
- Включає **пагінацію**, щоб на кожній сторінці відображалося лише 5 тем.
- Додається форма пошуку, яка дає змогу фільтрувати результати за назвою теми.

```python
class TopicListView(ListView):
    model = Topic
    paginate_by = 5
    queryset = Topic.objects.all()

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(TopicListView, self).get_context_data(**kwargs)

        name = self.request.GET.get("name", "")

        context["search_form"] = TopicSearchForm(initial={"name": name})
        return context

    def get_queryset(self):
        form = TopicSearchForm(self.request.GET)
        queryset = super().get_queryset()

        if form.is_valid():
            name = form.cleaned_data.get("name")
            if name:
                queryset = queryset.filter(name__icontains=name)
        return queryset
```

Рис. 3.8. Клас TopicListView

Клас `TopicCreateView` **(Створення теми):**

- Використовує `CreateView` для створення нової теми.
- Після успішного створення нової теми, користувача перенаправляє на сторінку зі списком усіх тем.

Клас `TopicUpdateView` **(Оновлення теми):**

- Використовує `UpdateView` для редагування вже існуючої теми.
- Після успішного оновлення користувач буде направлений на сторінку зі списком тем.

Клас `TopicDeleteView` **(Видалення теми):**

- Базується на `DeleteView` і дозволяє видаляти теми з бази даних.
- Після видалення користувача перенаправляє на список всіх тем.

```python
class TopicCreateView(LoginRequiredMixin, CreateView):
    model = Topic
    form_class = TopicForm
    success_url = reverse_lazy("newspaper:topics")


class TopicUpdateView(LoginRequiredMixin, UpdateView):
    model = Topic
    form_class = TopicForm
    success_url = reverse_lazy("newspaper:topics")


class TopicDeleteView(LoginRequiredMixin, DeleteView):
    model = Topic
    success_url = reverse_lazy("newspaper:topics")
```

Рис. 3.9. TopicCreateView, TopicUpdateView та TopicDeleteView

Клас `NewspaperListView` **(Список газет):**

- Використовує `ListView` для відображення списку газет.
- Так само, як і в інших списках, застосовується **пагінація**: по 5 газет на сторінку.
- Включає **форму пошуку** для фільтрації газет за назвою, а також **попереднє завантаження даних** (метод `prefetch_related`), що дозволяє ефективно отримувати пов'язані з газетами дані про теми і редакторів.

```python
class NewspaperListView(ListView):
    model = Newspaper
    paginate_by = 5
    queryset = Newspaper.objects.prefetch_related("topics").prefetch_related(
        "publishers"
    )

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(NewspaperListView, self).get_context_data(**kwargs)

        title = self.request.GET.get("title", "")

        context["search_form"] = NewspaperSearchForm(initial={"title": title})
        return context

    def get_queryset(self):
        form = NewspaperSearchForm(self.request.GET)
        queryset = super().get_queryset()

        if form.is_valid():
            title = form.cleaned_data.get("title")
            if title:
                queryset = queryset.filter(title__icontains=title)
        return queryset
```

Рис. 3.10. Клас NewspaperListView

Клас `NewspaperDetailView` **(Деталі газети):**

- Цей клас, що базується на `DetailView`, призначений для відображення детальної інформації про конкретну газету.

Клас `NewspaperCreateView` **(Створення газети):**

- Відповідає за створення нової газети, використовуючи `CreateView`.
- Після успішного додавання нової газети користувач буде перенаправлений на сторінку зі списком газет.

Клас `NewspaperDeleteView` **(Видалення газети):**

- Здійснює видалення обраної газети з бази даних через `DeleteView`.
- Після успішного видалення користувача перенаправляє на список газет.

Клас `NewspaperUpdateView` **(Оновлення газети):**

- Використовує `UpdateView` для редагування даних про газету.

- Після оновлення інформації про газету користувач буде перенаправлений на список газет.

```
153  class NewspaperDetailView(DetailView):
154      model = Newspaper
155
156
157  class NewspaperCreateView(LoginRequiredMixin, CreateView):
158      model = Newspaper
159      form_class = NewspaperForm
160      success_url = reverse_lazy("newspaper:newspapers")
161
162
163  class NewspaperDeleteView(LoginRequiredMixin, DeleteView):
164      model = Newspaper
165      success_url = reverse_lazy("newspaper:newspapers")
166
167
168  class NewspaperUpdateView(LoginRequiredMixin, UpdateView):
169      model = Newspaper
170      form_class = NewspaperForm
171      success_url = reverse_lazy("newspaper:newspapers")
172
```

Рис. 3.11. NewspaperDetailView, NewspaperCreateView, NewspaperDeleteView та NewspaperUpdateView відповідно

Основні принципи роботи та особливості:

- **Класові представлення (CBVs)**: Використовуються для реалізації стандартних операцій над даними (CRUD: створення, перегляд, оновлення, видалення) для кожної з моделей (редактори, теми, газети). Це дозволяє зменшити кількість коду, оскільки Django надає вже готові інструменти для таких операцій.
- **Форми пошуку**: Для кожного списку (редакторів, газет, тем) передбачено фільтрування результатів через параметри пошуку (наприклад, за іменем або заголовком). Ці форми дозволяють ефективно знаходити потрібні елементи в базі даних.

- **Аутентифікація**: Усі операції, що змінюють дані (створення, редагування, видалення), захищені через `LoginRequiredMixin`, що гарантує доступ лише для авторизованих користувачів.
- **Пагінація**: Для кожного списку (редакторів, тем, газет) використовується пагінація для обмеження кількості відображених елементів на сторінці. Це дозволяє зробити інтерфейс більш зручним та швидким.
- **Перенаправлення після операцій**: Після будь-якої операції (створення, редагування, видалення) користувача автоматично перенаправляють на відповідну сторінку, щоб зберегти зручність навігації.

Цей модуль надає набір функцій для адміністрування контенту на сайті. За допомогою представлених класів можна управляти редакторами, темами та газетами, надаючи користувачам або адміністраторам можливість створювати, переглядати, редагувати та видаляти відповідні записи. Система побудована за допомогою Django, що дозволяє швидко реалізувати ці функції через стандартні класові представлення.

Django надає широкий набір інструментів і бібліотек, які допомагають створювати форми для прийому вводу від користувачів сайту, а також обробляти та відповідати на цей ввід. Форми можуть використовуватися для збору даних, таких як інформація користувача, запити, відгуки, або для здійснення різноманітних операцій, таких як реєстрація користувачів, авторизація, створення або редагування записів у базі даних.

Опис форм у файлі forms.py

1. **Форма** `NewspaperForm`**:**

   Це форма для створення та редагування записів про газети.

   Вона базується на `ModelForm` і включає два поля для зв'язку з іншими моделями:

- `topics`: Поле для вибору однієї або кількох тем газети. Використовується поле `ModelMultipleChoiceField`, що дозволяє вибирати кілька варіантів через чекбокси (відображаються через `CheckboxSelectMultiple`).

```python
class NewspaperForm(forms.ModelForm):
    topics = forms.ModelMultipleChoiceField(
        queryset=Topic.objects.all(),
        widget=forms.CheckboxSelectMultiple()
    )
    publishers = forms.ModelMultipleChoiceField(
        queryset=Redactor.objects.all(),
        widget=forms.CheckboxSelectMultiple()
    )

    class Meta:
        model = Newspaper
        fields = "__all__"


class NewspaperSearchForm(forms.Form):
    title = forms.CharField(
        max_length=255,
        required=False,
        label="",
        widget=forms.TextInput(attrs={"placeholder": "Search by title"}),
    )


class RedactorCreationForm(UserCreationForm):
    class Meta(UserCreationForm.Meta):
        model = Redactor
        fields = UserCreationForm.Meta.fields + ("years_of_experience",)
```

Рис. 3.12. NewspaperForm, NewspaperSearchForm та NewspaperCreationForm

- `publishers`: Поле для вибору редакторів, які є авторами газети. Також використовує `ModelMultipleChoiceField` і `CheckboxSelectMultiple` для дозволу вибору кількох редакторів.

  В `Meta` класі вказано, що форма використовує модель `Newspaper` і включає всі поля цієї моделі (`"__all__"`), що дозволяє працювати з усіма атрибутами газети.

2. **Форма** `NewspaperSearchForm`**:**

   Це форма для пошуку газет за назвою.

   Вона використовує `CharField` для вводу тексту, де можна ввести частину назви газети.

   Атрибут `required=False` дозволяє зробити поле необов'язковим, а `widget=forms.TextInput(attrs={"placeholder": "Search by title"})` додає підказку у вигляді тексту в полі вводу (placeholder).

Ця форма дозволяє користувачеві шукати газети, вводячи назву газети, без необхідності вказувати точний запит.

3. **Форма** `RedactorCreationForm`**:**

   Ця форма базується на стандартній формі `UserCreationForm` з додаванням одного поля для створення редактора.

   Вона використовує ті ж самі поля, що й `UserCreationForm`, але додає поле `years_of_experience`, яке є специфічним для моделі `Redactor` і зберігає інформацію про досвід редактора.

   В класі `Meta` вказано, що ця форма працює з моделлю `Redactor` і містить усі стандартні поля для створення користувача, доповнені полем досвіду.

4. **Форма** `RedactorUpdateForm`**:**

   Це форма для редагування даних про редактора.

   Вона використовує `ModelForm` для моделі `Redactor` і включає лише три поля:

- `first_name`: ім'я редактора.
- `last_name`: прізвище редактора.
- `years_of_experience`: досвід редактора.

  Вона дозволяє редагувати основну інформацію редактора без зміни інших полів.

5. **Форма** `RedactorSearchForm`**:**

   Це форма для пошуку редакторів за ім'ям користувача.

   Вона містить одне поле `CharField`, яке дозволяє вводити частину імені користувача для пошуку.

   Визначена підказка `placeholder="Search by username"` допомагає користувачу зрозуміти, що саме він може шукати в цьому полі.

   Форма дозволяє здійснювати фільтрацію редакторів за їхніми іменами, що полегшує навігацію в списках редакторів.

6. **Форма** `TopicForm`**:**

Це форма для створення або редагування тем.

Вона є моделлю для форми, що базується на моделі `Topic`, і включає всі поля цієї моделі за допомогою `"__all__"`. Це дозволяє користувачеві додавати або редагувати тему з усіма її атрибутами.

7. **Форма** `TopicSearchForm`**:**

   Форма для пошуку тем за назвою.

   Вона містить одне текстове поле `CharField`, яке дозволяє здійснювати пошук за назвою теми.

   В поле вводу додається підказка у вигляді `placeholder="Search by name"`, щоб користувач зрозумів, що він може шукати за назвою теми.

Основні принципи роботи та особливості:

- **ModelForm**: Більшість форм базуються на `ModelForm`, що дозволяє автоматично створювати форми для моделей, беручи до уваги всі її поля та відповідні типи даних. Це значно зменшує обсяг коду, необхідний для створення форм.
- **Поле** `ModelMultipleChoiceField`: Це поле дозволяє вибирати кілька елементів, таких як теми чи редактори, використовуючи чекбокси. Це підвищує зручність взаємодії з користувачем при виборі кількох варіантів.
- **Атрибути віджетів**: Віджети форм, такі як `CheckboxSelectMultiple`, дозволяють представити дані в зручному вигляді, наприклад, через чекбокси для вибору кількох варіантів. Додаткові атрибути, як `placeholder`, надають користувачу інструкції щодо заповнення полів.
- **Пошукові форми**: Для кожного списку (газет, редакторів, тем) створена окрема форма для пошуку, що дозволяє користувачам фільтрувати результати за різними критеріями, такими як назва, ім'я користувача чи інші параметри. Це дозволяє швидше знаходити необхідні записи в базі даних.

Модуль `urls.py` в Django використовується для визначення URL-шляхів (маршрутів) на сайті та їхнього зв'язку з відповідними функціями чи класами представлення. Це дозволяє реалізувати навігацію по сайту, асоціюючи кожну сторінку або дію з певним URL, який викликає відповідну логіку в контролерах (представленнях). В даному випадку цей файл налаштовує маршрути для управління редакторами, темами та газетами на сайті.

```python
from django.urls import path

from newspaper.views import (
    index,
    RedactorListView,
    TopicListView,
    NewspaperListView,
    NewspaperCreateView,
    TopicCreateView,
    RedactorDetailView,
    NewspaperDetailView,
    RedactorCreateView,
    RedactorUpdateView,
    RedactorDeleteView,
    NewspaperDeleteView,
    NewspaperUpdateView, TopicUpdateView, TopicDeleteView,
)

urlpatterns = [
    path("", index, name="index"),
    path("redactors/", RedactorListView.as_view(), name="redactors"),
    path("redactors/create", RedactorCreateView.as_view(), name="redactor-create"),
    path("redactors/<int:pk>/", RedactorDetailView.as_view(), name="redactor-detail"),
    path(
        "redactors/<int:pk>/update",
        RedactorUpdateView.as_view(),
        name="redactor-update",
    ),
    path(
        "redactors/<int:pk>/delete",
        RedactorDeleteView.as_view(),
        name="redactor-delete",
    ),
    path("topics/", TopicListView.as_view(), name="topics"),
    path("topics/create/", TopicCreateView.as_view(), name="topic-create"),
    path("topics/<int:pk>/update/", TopicUpdateView.as_view(), name="topic-update"),
    path("topics/<int:pk>/delete/", TopicDeleteView.as_view(), name="topic-delete"),
    path("newspapers/", NewspaperListView.as_view(), name="newspapers"),
    path("newspapers/<int:pk>", NewspaperDetailView.as_view(), name="newspaper-detail"),
    path("newspapers/create/", NewspaperCreateView.as_view(), name="newspaper-create"),
    path(
        "newspapers/<int:pk>/delete",
        NewspaperDeleteView.as_view(),
        name="newspaper-delete",
    ),
    path(
        "newspapers/<int:pk>/update",
        NewspaperUpdateView.as_view(),
        name="newspaper-update",
    ),
]
app_name = "newspaper"

```

Рис. 3.13. Модуль urls.py

Особливості:

- `<int:pk>`: Цей параметр вказує на використання первинного ключа (`pk`) для ідентифікації конкретних записів (редакторів, тем або газет). Це дозволяє працювати з конкретними об'єктами, отримувати їх дані або виконувати зміни.
- `as_view()`: Цей метод використовуються для виклику класових представлень (CBVs), перетворюючи їх на функціональні представлення для роботи з URL-адресами. Це дозволяє легко визначати представлення, що реалізують CRUD-операції для моделей.
- `name`: Кожен шлях має унікальне ім'я (`name`), що дозволяє зручно посилатися на ці шляхи в шаблонах, за допомогою тегу `{% url %}`, що забезпечує гнучкість у зміні URL-адрес без зміни коду шаблонів.

Модуль `admin.py` в Django використовується для реєстрації моделей, які повинні бути доступні для управління через адміністративний інтерфейс. За допомогою цього файлу можна визначити, які моделі будуть відображатися в адміністративній панелі, а також налаштувати їхній вигляд та функціональність. Це дає можливість адміністраторам ефективно керувати даними в базі через веб-інтерфейс без необхідності писати SQL-запити.

Особливості:

- **Реєстрація моделей**: Моделі `Redactor`, `Topic` та `Newspaper` реєструються без налаштувань спеціальних параметрів, що означає, що Django автоматично відображатиме всі поля цих моделей в адміністративному інтерфейсі.
- **Доступність через адмін-панель**: Після реєстрації моделей в адмін-панелі з'являються форми для додавання, редагування та видалення записів цих моделей. Це дозволяє адміністраторам зручно керувати контентом сайту без необхідності втручання в базу даних вручну.

Цей файл дозволяє зареєструвати основні моделі сайту в адміністративному інтерфейсі Django. Він забезпечує просте управління редакторами, темами та газетами через вбудовану панель адміністратора, де

адміністратори можуть здійснювати CRUD-операції (створення, читання, оновлення, видалення) над даними, не вдаючись до написання SQL-запитів або використання сторонніх інструментів для роботи з базою даних.

Якщо в майбутньому потрібно буде налаштувати вигляд або поведінку адміністративного інтерфейсу для цих моделей (наприклад, додати фільтрацію, пошук чи спеціальні поля), можна буде створити власні класи адміністраторів і зареєструвати їх через `admin.site.register()`.

Модуль `apps.py` в Django визначає конфігурацію додатку. Він надає можливість налаштувати різні параметри додатку, такі як ім'я додатку, параметри бази даних, використовувані моделі та інші важливі аспекти функціонування додатку. Кожен Django-додаток має свій конфігураційний клас, який містить настройки для конкретного додатку.

newspaper > apps.py > ...

```python
from django.apps import AppConfig


class NewspaperConfig(AppConfig):
    default_auto_field = "django.db.models.BigAutoField"
    name = "newspaper"

```

Рис. 3.14. Модуль apps.py

- `class NewspaperConfig(AppConfig)::`

Оголошення класу `NewspaperConfig`, який успадковує від `AppConfig`. Цей клас відповідає за конфігурацію додатку `newspaper`.

- `default_auto_field = "django.db.models.BigAutoField":`

Це налаштування визначає тип автоматичного поля для кожної нової моделі, що буде створюватися в додатку. В даному випадку, встановлюється використання `BigAutoField`, який є полем для автоматичного інкрементованого ідентифікатора, але з більшим діапазоном значень, ніж стандартне поле `AutoField`. Це важливо, якщо додаток потребує великої

кількості записів, оскільки цей тип поля дозволяє зберігати більші значення для унікальних ідентифікаторів.

- `name = "newspaper"`:

Це ім'я додатку. Воно вказує Django, що цей клас належить додатку з назвою `newspaper`. Це ім'я використовується при реєстрації додатку в проекті Django та дозволяє уникнути конфліктів між різними додатками.

Призначення:

Модуль `apps.py` використовується для налаштування параметрів самого додатку в рамках Django-проекту. В основному, конфігурація додатку визначає:

- **Ім'я додатку**: Зазначене через параметр `name`. Це ім'я буде використовуватися для реєстрації додатку в проекті.
- **Тип автоматичних полів**: Через параметр `default_auto_field` налаштовується тип для автоматично згенерованих полів у моделях. У цьому випадку обрано `BigAutoField`, що є кращим вибором для додатків, де передбачається велика кількість записів.

Особливості:

- **Тип автоматичного поля**: Вказівка на `BigAutoField` дозволяє Django використовувати більший тип чисел для ідентифікаторів, що допомагає уникнути проблем з обмеженням діапазону значень у великих додатках або додатках з великою кількістю записів.
- **Модуль для налаштування додатку**: Цей файл може бути розширений для додаткової конфігурації додатку. Наприклад, можна налаштовувати сигнали, кешування або інші аспекти додатку, якщо це необхідно.

Модуль `apps.py` служить для налаштування конфігурації самого додатку в Django. Це дозволяє контролювати певні параметри роботи додатку, такі як тип автоматичних полів, а також гарантує правильне ідентифікування і реєстрацію додатку в Django-проекті.

3.3. Розробка інтерфейсу користувача

Одним із ключових елементів створення веб-додатків є забезпечення зв’язку між серверною логікою (бекендом) і візуальним відображенням (фронтендом). Django, будучи потужним веб-фреймворком, пропонує інтегровані інструменти для створення інтерфейсів користувача через використання системи шаблонів.

Шаблонний двигун Django

У Django функція `render` є основним інструментом для генерації HTML-відповідей із використанням шаблонів. Ця функція дозволяє розробникам передавати дані з бекенду в HTML-шаблони та відображати динамічний вміст користувачу.

Принцип роботи render

1. **Шаблони та їх розташування**:

   Шаблони зберігаються в папці `templates` у кореневій директорії проєкту або додатків. Ця структура дозволяє відокремлювати HTML-код від серверної логіки.

   Шаблони є статичними файлами HTML із вбудованою логікою, що реалізується за допомогою системи шаблонізації.

2. **Передача даних у шаблони**:

   Функція `render` приймає три основні параметри: запит (`request`), назву шаблону (`template_name`) і контекст (`context`).

   Контекст — це словник, який містить дані, що будуть доступні у шаблоні. Ці дані можуть включати текст, числа, списки, об’єкти або навіть результати складних обчислень.

3. **Динамічний контент у шаблонах**:

   HTML-шаблони можуть використовувати плейсхолдери для відображення змінних, переданих через контекст. Наприклад, `{% for item in items %}` створює динамічний список на основі даних із бекенду.

Використовуючи логіку шаблонів (цикли, умовні оператори), можна відображати персоналізований або змінний контент залежно від дій користувача.

Особливості шаблонів Django

1. **Наслідування шаблонів**:

- Django підтримує базові шаблони, які визначають загальний макет сайту (наприклад, навігаційні панелі, шапка, футер).
- Дочірні шаблони можуть розширювати базові, визначаючи специфічний вміст для конкретних сторінок через блоки (`{% block content %} ... {% endblock %}`).

2. **Безпека**:

- Вміст, відображений у шаблонах, автоматично ескейпиться для захисту від атак, таких як XSS (Cross-Site Scripting).
- Django дозволяє вручну управляти ескейпінгом через фільтри (`|safe`).
- **Шаблонізація через **`{{ ... }}` і `{% ... %}`:

  `{{ variable }}` — використовується для відображення значень змінних.

  `{% statement %}` — виконує логічні операції, як-от цикли чи умовні конструкції.

- Окрім динамічних шаблонів, Django підтримує роботу зі статичними файлами (CSS, JavaScript, зображення):

  `STATICFILES_DIRS` — змінна у файлі налаштувань, яка вказує, де зберігаються статичні файли.

  У шаблонах статичні файли підключаються через тег `{% static 'path/to/file' %}`.

Контекст і функція `render`

Ключова роль функції `render` — це динамічне створення сторінок із передачею даних.

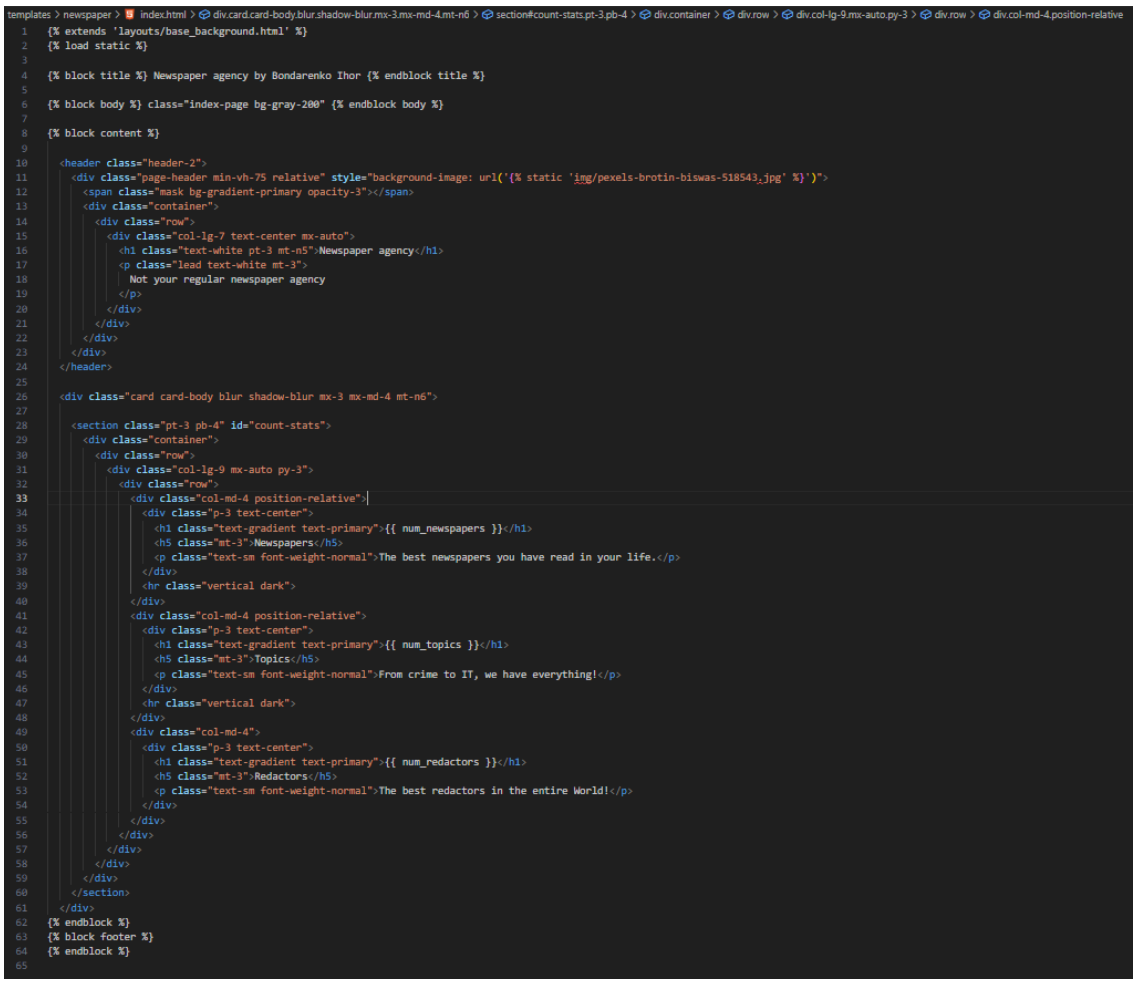

Рис. 3.15. Приклад шаблону головної сторінки

Інтеграція фронтенду з серверною логікою

1. **Валідація та форми**:

   Django автоматизує обробку форм через систему `forms.py`, дозволяючи інтегрувати їх у шаблони для збору даних.

2. **Асинхронність**:

   Для інтерактивності можна використовувати JavaScript або фронтенд-фреймворки разом із API, створеними через Django REST Framework.

3. **Роль `static` та `media` файлів**:

   Статичні файли використовуються для стилізації, а медіафайли — для завантаження і відображення користувацьких даних.

Переваги використання шаблонів у Django

1. **Чіткий поділ логіки та відображення.** HTML-код ізольований від Python-коду, що спрощує підтримку.

2. **Швидка розробка.** Django надає потужні інструменти для швидкого створення інтерфейсу.
3. **Універсальність.** Система шаблонів підтримує як базові, так і складні проєкти.
4. **Розширюваність.** Шаблони можна легко інтегрувати із сучасними фреймворками (React, Vue).

Значить, розробка фронтенду в Django базується на потужному механізмі шаблонізації, який поєднує гнучкість, безпеку та продуктивність.

3.4. Інструкція адміністратору

Цей пункт містить інструкції для користувача із роллю адміністратора.

Як запустити цей проєкт локально

1. **Склонуйте репозиторій на свій локальний комп'ютер**:

   git clone https://github.com/your_username/newspaper-agency-management.git

2. **Перейдіть до директорії проєкту**:

   cd newspaper-agency-management

3. **Створіть віртуальне середовище та активуйте його** (необов'язково, але рекомендовано):

   python -m venv venv
   source venv/bin/activate

4. **Встановіть залежності**:

   pip install -r requirements.txt

5. **Застосуйте міграції**:

   python manage.py migrate

6. **Створіть суперкористувача для доступу до адмін-панелі**:

   python manage.py createsuperuser

7. **Запустіть сервер розробки**:

   python manage.py runserver

8. **Відкрийте додаток у веббраузері за адресою**: http://localhost:8000/.

Використання

- **Адмін-панель**: Отримайте доступ до Django адмін-панелі за адресою http://localhost:8000/admin/ для керування редакторами, темами та газетами.
- **Керування редакторами**: Додавайте, редагуйте або видаляйте редакторів через адмін-панель. Слідкуйте за їхніми іменами користувачів, іменами та кількістю років досвіду.
- **Керування темами**: Створюйте, редагуйте або видаляйте теми через адмін-панель. Призначайте теми газетам, щоб ефективно категоризувати контент.
- **Публікація газет**: Додавайте нові газети в адмін-панелі. Вказуйте заголовок, зміст, дату публікації та призначайте редакторів і теми до кожної газети.
- **Призначення декількох тем**: При додаванні або редагуванні газети ви можете призначати їй декілька тем, щоб забезпечити повну категоризацію вашого контенту.

3.5. Практичний приклад роботи веб-додатку з точки зору користувача

Після реєстрації користувача адміністратором, йому потрібно натиснути на кнопку Login та ввести свій логін та пароль.

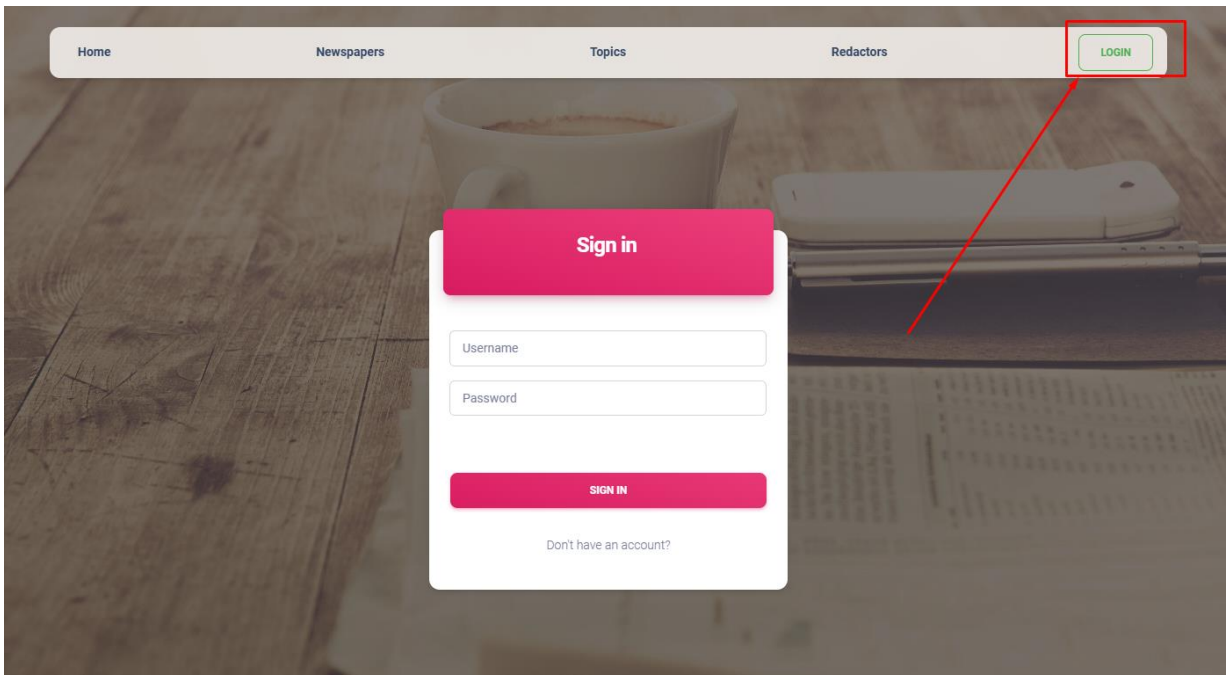

Рис. 3.16. Сторінка Sign in

На сторінці My cabinet користувач може переглядати та змінювати свою персональну інформації, видалити свій акаунт та бачити всі створені ним газети.

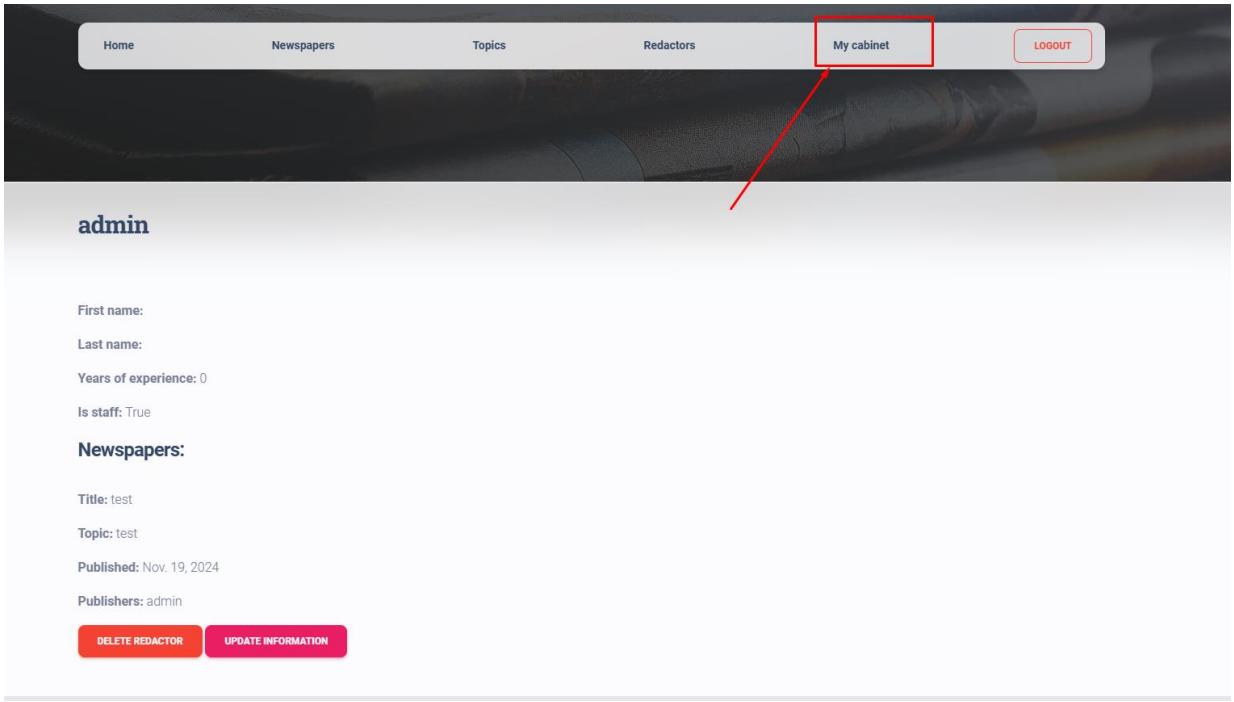

Рис. 3.17. Сторінка My cabinet

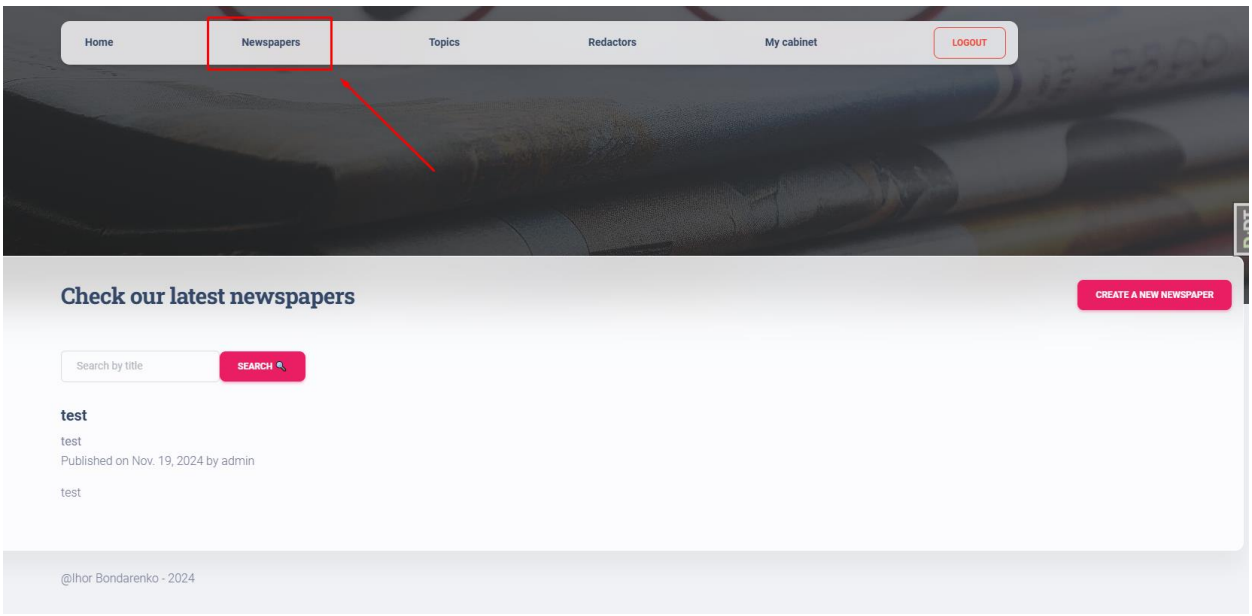

Рис. 3.18. Сторінка Newspapers

Сторінка Newspapers потрібна для перегляду всіх газет від всіх редакторів. За допомогою кнопки Create new newspaper юзер може створити нові газети при цьому вказати заголовок, вміст, вибрати тему та вказати всіх редакторів, що працювали над нею.

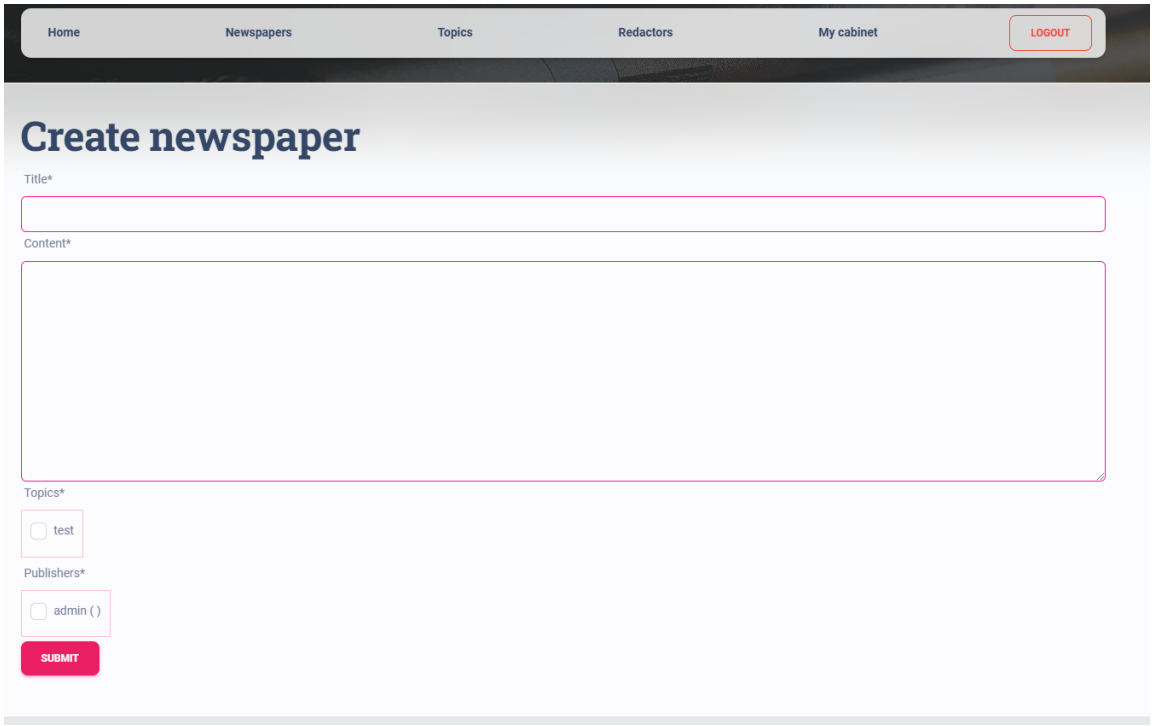

Рис. 3.19. Створення нової газети

Сторінка Topics надає змогу додати нові теми для газет та переглядати, видаляти та редагувати наявні.

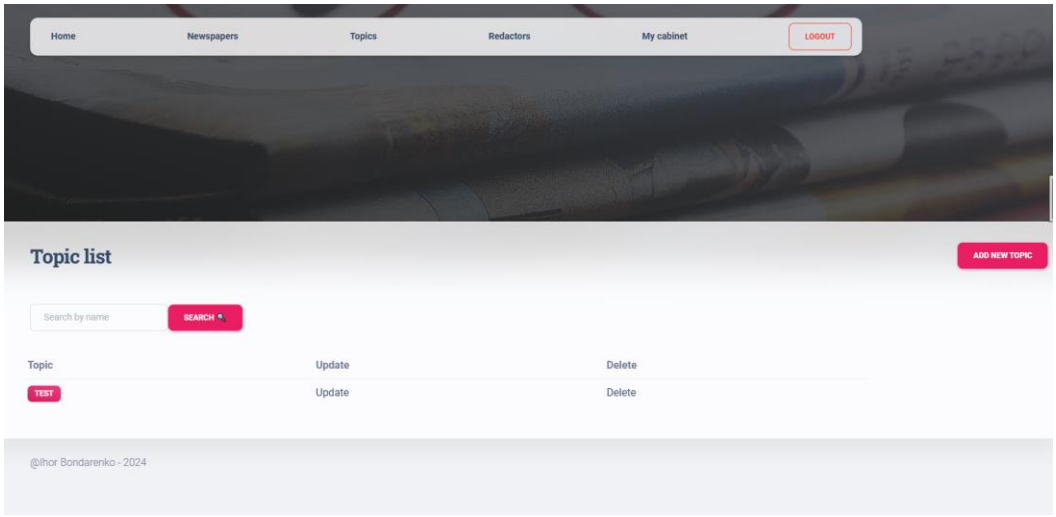

Рис 3.20. Сторінка Topics

3.6. Висновки до 3 розділу

У цьому розділі було представлено практичний приклад створення веб-додатку на Python з використанням SQLite для управління даними абітурієнтів у навчальному закладі. Всі етапи розробки, починаючи від архітектури додатку до розробки інтерфейсу користувача та програмування backend частини, були детально описані, що дозволяє чітко уявити процес створення подібного веб-додатку.

Розробка моделей даних та інтеграція з базою даних через SQLite стала основою для створення надійної та ефективної системи зберігання інформації. Вибір фреймворку Django забезпечив зручність і швидкість розробки завдяки своїм вбудованим можливостям для управління базами даних, створення інтерфейсу адміністратора та забезпечення безпеки даних.

Особливу увагу було приділено питанням забезпечення безпеки, оптимізації роботи з базою даних та створенню зручного інтерфейсу для кінцевих користувачів. У результаті розроблений додаток є надійним інструментом для ефективного управління даними абітурієнтів, що може бути використано в реальних освітніх установах.

Технічні рішення, обрані для реалізації цього веб-додатку, продемонстрували можливості Python, Django і SQLite для розробки легких і продуктивних веб-додатків, що відповідають вимогам сучасного освітнього процесу. Завдяки цьому дослідженню ми змогли наочно показати, як теоретичні знання у галузі веб-розробки можуть бути ефективно застосовані на практиці для вирішення реальних завдань.

4. ВИМОГИ ДО ФАХІВЦІВ З КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБ-ДОДАТКІВ НА PYTHON

У сучасному інформаційному суспільстві критично важливим є забезпечення підготовки фахівців, які здатні професійно створювати веб-додатки на основі Python і Django. Розробка таких систем вимагає від спеціалістів високого рівня технічної компетенції, знання актуальних інструментів та дотримання передових практик. Такі спеціалісти повинні володіти глибокими знаннями в галузі програмування, систем управління базами даних, інтеграції серверних і клієнтських компонентів, а також бути здатними розробляти масштабовані, безпечні й ефективні рішення.

Особливу увагу слід приділити вмінню налаштовувати середовище розробки, інтегрувати серверні компоненти з інтерфейсом користувача, застосовувати міграції баз даних для структурування та збереження інформації, а також створювати динамічні веб-інтерфейси з використанням шаблонів Django. Ефективна робота у цій галузі передбачає також знання принципів безпеки веб-додатків, включно із захистом від XSS-атак, SQL-ін'єкцій та налаштуванням авторизації й автентифікації.

4.1. Кваліфікаційні вимоги до фахівців у галузі веб-розробки

У сучасних умовах розвитку інформаційних технологій веб-розробка стає одним із ключових напрямків у сфері комп'ютерних технологій. Фахівці в цій галузі повинні володіти глибокими знаннями та практичними навичками, щоб створювати ефективні, надійні та масштабовані веб-додатки. Основні кваліфікаційні вимоги до таких спеціалістів включають технічну компетентність, здатність працювати в команді, а також розуміння сучасних тенденцій у веб-розробці.

Перш за все, розробник має демонструвати сильне розуміння основних концепцій програмування. Це включає знання алгоритмів, структур даних та принципів об'єктно-орієнтованого програмування. Особлива увага приділяється знанню мови програмування Python, яка є однією з найпопулярніших у веб-

розробці завдяки своїй універсальності, легкості у вивченні та широкій підтримці спільноти.

Кваліфікований фахівець також має володіти знаннями у сфері веб-технологій, таких як HTML, CSS і JavaScript. Ці технології формують основу взаємодії користувача з веб-додатком. Крім того, розуміння роботи фреймворків (наприклад, Django або Flask) є обов'язковим для розробки серверної частини додатків. Також важливими є знання про бази даних, особливо реляційні (наприклад, PostgreSQL або MySQL), а також досвід роботи з ORM-інструментами.

Фахівець з веб-розробки повинен вміти працювати з системами контролю версій, такими як Git. Це забезпечує ефективну роботу в команді, дозволяє відслідковувати зміни в коді та інтегрувати оновлення, створені різними розробниками.

Не менш важливою є здатність до аналітичного мислення, яка дозволяє вирішувати складні технічні завдання, оптимізувати код і забезпечувати високу продуктивність додатків. Розробник повинен бути готовий до постійного навчання, оскільки технології та інструменти у веб-розробці стрімко змінюються.

Окрім технічних навичок, веб-розробник має володіти комунікативними здібностями. Це включає здатність ефективно обговорювати технічні питання з колегами, а також взаємодіяти з клієнтами для уточнення їхніх вимог. Також важливо мати навички управління часом, щоб дотримуватися встановлених термінів реалізації проєктів.

Таким чином, кваліфікаційні вимоги до веб-розробника формуються на основі технічних знань, професійних навичок, здатності до самонавчання і ефективної взаємодії у команді. Ці компетенції є основою для успішної кар'єри у сфері веб-розробки, а також забезпечують високий рівень розробки програмного забезпечення.

4.2. Технології, які має опанувати розробник

Ефективна розробка веб-додатків на Python вимагає від фахівця володіння широким спектром технологій, що забезпечують повний цикл створення веб-продуктів — від проектування інтерфейсу до налаштування серверної логіки та управління базами даних. Технології, які повинен опанувати розробник, можна поділити на декілька основних категорій: фронтенд, бекенд, бази даних, інструменти розробки та супутні технології.

Фронтенд-технології формують основу для створення користувацького інтерфейсу. Розробник повинен мати базові знання HTML, CSS та JavaScript. HTML використовується для структурування веб-сторінок, CSS забезпечує їхнє стилізування, а JavaScript додає інтерактивність. Для спрощення роботи з JavaScript розробник може використовувати сучасні бібліотеки та фреймворки, такі як React або Vue.js. У цьому контексті знання принципів адаптивного дизайну також є обов’язковими, оскільки сучасні додатки повинні коректно відображатися на різних пристроях.

Бекенд-технології включають інструменти та фреймворки для реалізації серверної логіки додатка. Django є одним із найбільш популярних фреймворків для Python, який забезпечує швидку розробку, безпеку та масштабованість. Django надає вбудовані функції для роботи з базами даних, аутентифікації, формування API та інших завдань. Крім Django, розробник може використовувати Flask для легших додатків, які потребують більшої гнучкості.

Бази даних є невіддільною частиною більшості веб-додатків. Розробник повинен володіти знаннями про реляційні бази даних, такі як PostgreSQL або MySQL, а також розуміти основи роботи з ORM (Object-Relational Mapping), які дозволяють працювати з базами даних через Python-код. У Django ORM є вбудованим компонентом, що спрощує створення запитів та управління даними. Для специфічних завдань можуть знадобитися нереляційні бази даних, наприклад MongoDB.

Інструменти розробки допомагають оптимізувати роботу розробника. Серед них важливу роль відіграє система контролю версій Git, яка дозволяє

управляти змінами у коді та працювати у команді. Крім того, знання середовищ розробки (IDE), таких як PyCharm або Visual Studio Code, дозволяє збільшити продуктивність за рахунок використання інтегрованих функцій для налагодження, тестування та оптимізації коду.

Супутні технології включають інструменти для створення API, такі як Django REST Framework, який дозволяє розробляти інтерфейси для обміну даними між сервером та клієнтом. Крім того, важливим є володіння основами DevOps-практик, наприклад роботою з Docker для контейнеризації додатків, а також налаштування серверів для розгортання веб-додатків. Знання основ безпеки, таких як захист від SQL-ін’єкцій або XSS-атак, є критично важливими для створення безпечного програмного забезпечення.

Опанування цих технологій дозволяє розробнику створювати високоякісні веб-додатки, які відповідають сучасним вимогам зручності, функціональності та безпеки. Широкий набір технічних інструментів у поєднанні з вмінням їх інтегрувати є невід'ємною складовою професійної компетенції веб-розробника.

4.3. Особливості професійної підготовки фахівців

Професійна підготовка фахівців із розробки веб-додатків на Python є складним процесом, що поєднує теоретичні знання та практичні навички. Вона базується на сучасних освітніх підходах, які враховують швидкі темпи розвитку технологій та необхідність адаптації до динамічного IT-середовища. У цьому розділі розглянуто ключові аспекти навчання, які забезпечують формування висококваліфікованих веб-розробників.

Перш за все, програма підготовки повинна охоплювати базові концепції програмування, включаючи знання алгоритмів і структур даних, а також основи мови Python. Ці знання створюють фундамент для розуміння більш складних технологій і інструментів. Особлива увага приділяється принципам об’єктно-орієнтованого програмування (ООП), які широко використовуються у фреймворку Django.

Далі, важливим компонентом навчання є ознайомлення з методологіями розробки програмного забезпечення, такими як Agile чи Scrum. Розробники

мають розуміти процеси управління проектами, комунікації у команді та практики спільної роботи з використанням інструментів на кшталт GitHub або GitLab. Це забезпечує їх готовність працювати у реальних умовах розробки.

Особливості професійної підготовки передбачають інтеграцію теорії з практикою. Під час навчання студенти працюють над реальними проектами, що включають всі етапи створення веб-додатку: від постановки завдання до розгортання продукту на сервері. Наприклад, вони можуть створювати веб-додатки з використанням Django, налаштовувати бази даних, розробляти REST API та інтегрувати фронтенд із бекендом. Такий підхід дозволяє формувати практичні навички, які відповідають потребам ринку праці.

Ще одним важливим аспектом є навчання основам тестування програмного забезпечення. Уміння писати модульні та інтеграційні тести є ключовою навичкою, яка дозволяє забезпечити якість і стабільність веб-додатків. У Django студенти можуть використовувати вбудовані інструменти для тестування, що полегшує навчальний процес.

Підготовка також включає вивчення основ безпеки веб-додатків. Студенти мають знати про потенційні загрози, такі як SQL-ін’єкції чи XSS-атаки, та способи їхнього уникнення. Розуміння принципів роботи SSL/TLS, а також методів шифрування даних є обов’язковим для забезпечення захищеності додатків.

На завершення, фахівці з веб-розробки повинні бути навчені працювати з інфраструктурними інструментами, такими як Docker для контейнеризації або платформи хмарного хостингу для розгортання додатків. Це дає змогу ефективно керувати процесами доставки та підтримки програмного забезпечення.

Таким чином, професійна підготовка веб-розробників повинна бути багатогранною, інтегруючи як технічні аспекти, так і навички командної роботи, адаптивності та аналітичного мислення. Такий підхід дозволяє випускникам навчальних програм бути готовими до викликів сучасного ринку праці в сфері інформаційних технологій.

5. МЕТОДИЧНІ ВКАЗІВКИ ДО ПРОВЕДЕННЯ ФАКУЛЬТАТИВНОГО ЗАНЯТТЯ З ТЕМИ «СТВОРЕННЯ ВЕБ-ДОДАТКІВ НА PYTHON З ВИКОРИСТАННЯМ БАЗИ ДАНИХ» ДИСЦИПЛІНИ «МОБІЛЬНІ ТА WEB ПЛАТФОРМИ» ДЛЯ ПІДГОТОВКИ ФАХІВЦІВ З КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ.

В умовах швидкого розвитку інформаційних технологій в освітньому процесі стає все більш важливим використання новітніх методів для управління даними. Однією з ключових задач є організація та обробка даних про абітурієнтів у навчальних закладах, що вимагає ефективних підходів для зберігання та доступу до інформації.

Даний факультатив спрямований на розвиток практичних навичок студентів спеціальності "Професійна освіта (Цифрові технології)" у створенні веб-додатків. Студенти ознайомляться з процесом розробки веб-системи, яка дозволяє ефективно організовувати й обробляти дані абітурієнтів за допомогою Python та бази даних SQLite. У рамках заняття студентам буде надано практичний досвід використання фреймворку Django для створення web-сервісів, що взаємодіють з базами даних, та засвоєння основ захисту даних і налаштування системи.

Кожен етап заняття охоплює важливі аспекти створення веб-додатків. Студенти починають з аналізу вихідних вимог до веб-системи для обробки даних абітурієнтів. На наступному етапі вони освоюють проектування бази даних та створення відповідних моделей для зберігання інформації. Після цього студенти приступають до розробки серверної частини додатку, де вони навчаються працювати з фреймворком Django, створюючи API для взаємодії з базою даних SQLite.

Наступним кроком є розробка інтерфейсу користувача, який дозволить адміністраторам зручно працювати з даними абітурієнтів. Особливу увагу буде приділено тому, як організувати зручну взаємодію з системою, використовуючи шаблони Django для створення динамічних веб-сторінок.

Для підвищення мотивації та залучення студентів до активної роботи, буде розроблено спеціальні завдання та кейс-сценарії, що дозволяють побачити практичне застосування отриманих навичок у реальних ситуаціях. Це допоможе студентам краще зрозуміти значення веб-розробки та основи управління даними в освітніх установах.

Підсумкове оцінювання результатів навчання буде здійснюватися через виконання практичних завдань, тестів та презентацію результатів розроблених веб-додатків, що дозволить перевірити рівень засвоєння матеріалу та здатність до ефективного застосування знань.

Таким чином, методичні вказівки допоможуть організувати навчальний процес, надаючи студентам можливість практично освоїти технології веб-розробки та управління даними в сучасних інформаційних системах.

5.1. Вихідні дані:

Далі наведено вихідні дані роботи:

1. Тип навчального закладу: заклад вищої освіти.
2. Освітній та освітньо-кваліфікаційний рівні: бакалавр.
3. Назва спеціальності: "Професійна освіта (Цифрові технології)".
4. Стратегічні цілі підготовки фахівця стосовно теми факультативного заняття: формування у студентів навичок проектування, розробки та впровадження веб-додатків за допомогою Python і баз даних, що забезпечує ефективне управління інформацією та підтримку процесів автоматизації в реальних IT-проектах.
5. Назва дисципліни, з якої розроблятиметься практичне заняття: Мобільні та Web платформи.
6. Дидактичні цілі навчальної дисципліни, з якої проводитиметься практична робота:

a. Відшуковувати, обробляти, аналізувати та оцінювати інформацію, що стосується професійної діяльності, користуватися спеціалізованим програмним забезпеченням та сучасними засобами зберігання та обробки інформації.

b. Уміти проектувати і реалізувати навчальні/розвивальні проєкти.

c. Уміти обирати і застосовувати необхідне устаткування, інструменти та методи для вирішення типових складних завдань в галузі цифрових технологій.

5.2. Постановка цілей факультативного заняття.

Таблиця 5.1 – Оперативні цілі вивчення теми

Цілі факультативного заняття	**Цілі формування різних рівнів засвоєння навчального матеріалу**	**Умови досягнення**	**Результат у вигляді дій студентів**
Розуміння базових принципів створення веб-додатків на Python із використанням бази даних SQLite.	Теоретичний рівень: визначати основні елементи веб-додатку, пояснювати взаємодію між Python та SQLite.	Лекційне пояснення, презентація, інтерактивне обговорення.	Студенти здатні пояснити структуру веб-додатку та визначити його ключові компоненти.
Освоєння основних етапів розробки backend частини веб-додатку.	Продуктивно-практичний рівень: розробка моделей даних, налаштування взаємодії з базою даних через ORM Django.	Практична демонстрація, супроводжувані вправи, робота з прикладами коду.	Студенти здатні створити моделі даних та налаштувати їх інтеграцію з базою SQLite.
Формування навичок розробки інтерфейсу користувача для веб-додатків.	Продуктивно-практичний рівень: створення веб-сторінок, підключення шаблонів Django, реалізація форм для роботи з даними.	Практичні вправи під керівництвом викладача, обговорення можливих помилок та їх виправлення.	Студенти створюють базовий користувацький інтерфейс із функціоналом для взаємодії з даними.
Застосування отриманих знань у розробці веб-додатку з нуля.	Продуктивно-творчий рівень: розробка повноцінного веб-додатку з базою SQLite.	Самостійна проєктна робота з консультаціями викладача, використання прикладів з попередніх занять.	Студенти розробляють власний веб-додаток, що включає backend, базу даних та користувацький інтерфейс.
Демонстрація	Продуктивно-творчий	Організація	Студенти

Цілі факультативного заняття	Цілі формування різних рівнів засвоєння навчального матеріалу	Умови досягнення	Результат у вигляді дій студентів
готового веб-додатку та обговорення результатів.	рівень: аналіз та презентація функціональності веб-додатку.	презентації робіт, спільне обговорення досягнень та проблем.	демонструють розроблений продукт, пояснюють його функціональність та пропонують можливі покращення.

5.3. Перелік літературних джерел з теми.

- Python Software Foundation. *The Python Language Reference Manual*. Онлайн ресурс
- Holovaty A., Kaplan-Moss J. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2020.
- Owens M. *The Definitive Guide to SQLite*. Apress, 2019.
- Krug K. *Using SQLite*. O'Reilly Media, 2020.
- Flanagan D. *JavaScript: The Definitive Guide*. O'Reilly Media, 2020.
- Duckett J. *HTML and CSS: Design and Build Websites*. Wiley, 2014.
- Pollock D. *Beginning Python: Using Python 2.6 and Python 3.1*. Wrox, 2021.
- Matthes E. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press, 2019.
- В. В. Черемних, О. А. Скворцова. *Програмування на Python. Практичний курс*. Видавництво Національного технічного університету України, 2021.
- Методичні матеріали кафедри комп'ютерних наук та інформаційних технологій (локальні ресурси навчального закладу).
- Real Python — інтерактивні уроки Python та Django.
- Django Documentation — офіційна документація Django.
- Бушуев С. Д., Горбачук О. В. *Використання Python у створенні веб-додатків: досвід і перспективи*. *Сучасні інформаційні технології та інноваційні методи в освіті*, 2022.

- Smith J., Anderson P. *Modern Practices in Python Web Development*. *Journal of Software Engineering and Applications*, 2023.
- Sweigart A. *Automate the Boring Stuff with Python*. No Starch Press, 2020.
- Ziadé T. *Expert Python Programming*. Packt Publishing, 2019.

5.4. Конструювання дидактичних матеріалів.

Тема: *«Створення веб-додатків на Python з використанням бази даних»*.

Мета заняття: Формування у студентів практичних навичок розробки веб-додатків із використанням фреймворка Django та бази даних SQLite, а також розвиток умінь аналізувати, проєктувати й реалізовувати функціональні компоненти веб-додатків.

Компоненти структури навчального матеріалу:

1. **Вступ**:
   - Ознайомлення з метою заняття.
   - Огляд технологічного стеку (Python, Django, SQLite).
2. **Теоретична частина**:
   - Основи Python та Django для створення веб-додатків.
   - Архітектура MVC (Model-View-Controller) у Django.
   - Робота з SQLite: створення, читання, оновлення та видалення даних (CRUD-операції).
3. **Практична частина**:
   - Налаштування середовища розробки (установлення Python, Django, SQLite).
   - Розробка базової структури веб-додатку.
   - Створення моделі даних для управління інформацією.
   - Реалізація взаємодії з базою даних через ORM (Object-Relational Mapping).
   - Розробка інтерфейсу для відображення та управління даними.
4. **Рефлексія та підсумки**:
   - Аналіз виконаних завдань.

- Обговорення типових проблем і їх вирішення.
- Рекомендації для самостійного вивчення.

Характер зв’язків між компонентами:

Навчальний матеріал побудовано за принципом послідовності: вступ → теоретичні знання → практична реалізація → підсумковий аналіз. Кожен елемент логічно пов'язаний із попереднім і наступним, забезпечуючи цілісність навчального процесу.

Таблиця 5.2. План теми відповідно до її структури

Етап	Діяльність викладача	Діяльність студентів	Час
Вступ	Постановка мети заняття, пояснення значення теми	Слухають, ставлять уточнюючі запитання	10 хв.
Теоретична частина	Лекція з елементами демонстрації: концепції Django, SQLite, основи MVC	Записують, задають запитання, обговорюють	30 хв.
Практична частина	Пояснення завдання, демонстрація початкових кроків	Налаштовують середовище, виконують завдання	50 хв.
Рефлексія та підсумки	Аналіз результатів, обговорення помилок, відповіді на запитання	Демонструють роботи, беруть участь у дискусії	20 хв.

Очікувані результати:

- Студенти вміють налаштовувати середовище для розробки.
- Студенти можуть створювати веб-додаток із базовим функціоналом.
- Студенти отримують уявлення про інтеграцію бази даних із веб-додатком.

5.5. Аналіз базових умов навчання.

Таблиця 5.3. Аналіз базового матеріалу і способи актуалізації базових знань

Перелік базових понять, законів, способів дії	Назва дисциплін і тем, в яких формуються базові знання і дії	Способи (методи, форми, засоби) перевірки рівня сформованості базових знань і способів дій	Способи актуалізації або поповнення базових знань і способів дій
Основи програмування:	«Основи	Виконання	Демонстрація

Перелік базових понять, законів, способів дії	**Назва дисциплін і тем, в яких формуються базові знання і дії**	**Способи (методи, форми, засоби) перевірки рівня сформованості базових знань і способів дій**	**Способи актуалізації або поповнення базових знань і способів дій**
типи даних, функції, цикли, умовні оператори	програмування», тема «Базові конструкції Python»	тестових завдань, практичні вправи, аналіз вихідного коду	прикладів у Python, інтерактивні вправи для повторення
Основи роботи з базами даних: структура баз даних, операції SELECT, INSERT, UPDATE, DELETE	«Бази даних», тема «Реляційні бази даних і мови запитів SQL»	Виконання завдань на написання SQL-запитів	Обговорення реальних прикладів використання баз даних, створення простих запитів у SQLite
Архітектура веб-додатків: клієнт-серверна модель, принципи роботи HTTP	«Основи веб-технологій», тема «Клієнт-серверна архітектура та протокол HTTP»	Проведення тесту на теоретичне розуміння, аналіз структури запитів і відповідей HTTP	Розбір прикладів роботи протоколу HTTP у реальному часі через браузер та інструменти розробки
Основи фреймворків: поняття MVC (Model-View-Controller), структура проєктів Django	«Веб-розробка», тема «Фреймворки для створення веб-додатків»	Обговорення базових концепцій MVC, практичне завдання на ідентифікацію компонентів у структурі веб-додатку	Демонстрація прикладів організації коду в Django, аналіз базових шаблонів
Об’єктно-орієнтоване програмування: класи, об’єкти, методи	«Програмування», тема «Основи ООП»	Написання програм із використанням базових принципів ООП, аналіз виконаних завдань	Розбір практичних прикладів реалізації класів у Python
Основи роботи з ORM (Object-Relational	«Програмування баз даних», тема	Виконання практичних вправ:	Розробка невеликих

Перелік базових понять, законів, способів дії	Назва дисциплін і тем, в яких формуються базові знання і дії	Способи (методи, форми, засоби) перевірки рівня сформованості базових знань і способів дій	Способи актуалізації або поповнення базових знань і способів дій
Mapping): взаємодія між об’єктами Python і записами в базі даних	«ORM як інструмент роботи з базами даних»	написання коду для створення та оновлення записів у базі даних через ORM	прикладів у Django, які демонструють зв’язок між моделями та базою даних
HTML і CSS: основи верстки веб-сторінок	«Основи веб-технологій», тема «Мови розмітки і стилізації»	Перевірка знань через створення веб-сторінки за заданим макетом, тести	Повторення основ HTML/CSS шляхом аналізу простих веб-сторінок, інтерактивні вправи

Аналіз і висновки

Базові знання, необхідні для успішного виконання факультативного заняття, формуються на основі обов’язкових дисциплін, таких як «Основи програмування», «Бази даних» та «Основи веб-технологій». Актуалізація знань студентів здійснюється за допомогою інтерактивних прикладів, виконання практичних завдань і обговорення реальних кейсів.

5.6. Проектування мотиваційних технологій

Таблиця 5.4. Обрання методів мотивації навчальної діяльності

Вид і методи мотивації	Вступна мотивація	Підтримуюча (поточна) мотивація
Прийоми зовнішньої мотивації, віднесення до ситуації	«Шановні здобувачі освіти, сьогодні ми розпочнемо роботу над створенням реального веб-додатка з використанням фреймворка Django та бази даних SQLite. Це практичне завдання наближає вас до реальних проєктів, які ви зможете виконувати в професійній	«Зараз ми переходимо до критично важливого етапу – налаштування зв’язку між веб-додатком та базою даних. Ті, хто виконає цей етап якісно, отримають перевагу на наступному занятті й зможуть уникнути додаткових завдань удома».

	діяльності. Тож пропоную вам бути уважними, адже наприкінці заняття ми оцінимо ваші результати».	
Прийоми внутрішньої мотивації, віднесення до особистості	«Розробка веб-додатків є однією з найбільш затребуваних сфер IT. Ваші навички роботи з фреймворками, як-от Django, та базами даних, як SQLite, роблять вас конкурентними на ринку праці. Те, що ми вивчаємо сьогодні, – це фундамент для побудови сучасних та функціональних веб-додатків».	«Уявіть, що ви створюєте систему для реального замовника: від коректного проєктування бази даних залежатиме успіх вашого проєкту. Саме зараз ви розвиваєте навички, які дозволять вам створювати дійсно корисні програми».

5.7. Проектування технології формування орієнтовної основи діяльності

Таблиця 5.5. Способи формування ООД на факультативному занятті

№ задачі	**Методи, засоби формування ООД**	**Змістовні матеріали з формування ООД**
1	Налаштування середовища для розробки веб-додатка з базою даних	Пояснення алгоритму виконання:
1.1	Встановлення фреймворка Django	Пояснення дій щодо встановлення Django через менеджер пакетів pip
1.2	Створення нового проєкту Django	Пояснення дій щодо створення нового проєкту командою django-admin startproject project_name
1.3	Налаштування бази даних SQLite у файлі settings.py	Пояснення дій щодо налаштування підключення до бази даних SQLite у файлі конфігурації проекту
1.4	Створення та застосування міграцій	Пояснення дій щодо виконання команд python manage.py makemigrations та python manage.py migrate
1.5	Перевірка налаштування бази даних через адміністративну панель Django	Пояснення дій щодо доступу до адміністративної панелі та додавання записів до бази даних через інтерфейс адміністратора
2	Розробка функціональності	Пояснення алгоритму виконання:

	взаємодії з базою даних	
2.1	Створення моделі Django	Пояснення дій щодо визначення моделей у файлі models.py
2.2	Використання ORM Django для роботи з базою даних	Пояснення базових операцій ORM: створення, читання, оновлення та видалення записів у базі даних
2.3	Створення форми для введення даних	Пояснення дій щодо створення форми через клас forms.Form
2.4	Зв’язок форми з базою даних	Пояснення дій щодо збереження введених користувачем даних у базі даних за допомогою моделі
2.5	Виведення даних із бази в шаблонах Django	Пояснення дій щодо відображення даних у шаблоні через контекст та використання тегів Django

5.8. Проектування технології формування виконавчих дій

Таблиця 5.6. Способи формування виконавчих дій з теми

Рівні засвоєння навчального матеріалу теми заняття	**Форми**	**Методи**	**Засоби**
Ознайомчий рівень	Лекція з демонстрацією	Пояснювально-ілюстративний метод	Слайди з алгоритмом створення веб-додатка, приклади коду на Python та SQL, візуалізація взаємодії компонентів веб-додатка
Репродуктивний рівень	Практичне заняття	Покрокова інструкція, виконання завдань за алгоритмом	Шаблон Django-проєкту, приклади коду для створення моделей і форм, інструкція з налаштування SQLite
Продуктивно-	Виконання	Практичний метод, розбір	Вихідний код

Рівні засвоєння навчального матеріалу теми заняття	Форми	Методи	Засоби
практичний рівень	індивідуальних завдань	типових помилок	для самостійної доробки, завдання на створення та налаштування бази даних для конкретної навчальної задачі
Продуктивно-творчий рівень	Розробка мініпроєкту	Метод проєктів, евристичні методи	Завдання для створення функціонального модуля, шаблони тестових сценаріїв, доступ до документації Python, Django та SQLite
Контрольно-оцінювальний рівень	Захист виконаного проєкту	Метод оцінювання результатів роботи, самостійне тестування, взаємооцінювання	Критерії оцінювання (функціональність, дизайн, продуктивність), тестові дані для перевірки, середовище розробки з налаштованою базою

5.9. Проектування контрольних дій з теми.

Таблиця 5.7. – Засоби контролю з теми факультативного заняття

Рівні засвоєння навчального матеріалу теми заняття	Форми	Методи	Засоби
Ознайомчий рівень	Фронтальне	Усне опитування,	Тестові питання,

Рівні засвоєння навчального матеріалу теми заняття	Форми	Методи	Засоби
	опитування	тестовий контроль	зокрема на знання теорії про Python, SQLite, Django, основи створення веб-додатків
Репродуктивний рівень	Перевірка виконання практичних завдань	Контроль за виконанням алгоритмів, аналіз помилок	Задачі з шаблонним кодом, перевірка роботи SQLite-запитів, перевірка налаштування бази даних
Продуктивно-практичний рівень	Захист виконаного завдання	Практичний контроль	Аналіз виконання завдань (створення моделей, форм, функціональних модулів), лог перевірки роботи
Продуктивно-творчий рівень	Захист мініпроєкту	Метод проєктів, самоаналіз	Розгляд функціонального прототипу, його дизайну та продуктивності, аналіз тестових сценаріїв
Контрольно-оцінювальний рівень	Оцінювання результатів діяльності	Самооцінювання, взаємооцінювання, тестування	Критерії оцінювання (функціональність, точність даних, дизайн), порівняння з реальними прикладами

5.10. Розробка сценарію факультативного заняття.

Тип заняття: практично-теоретичне

Тривалість заняття: 2 години

Таблиця 5.8 – Сценарій навчання з теми заняття

№ п/п	Структурні елементи уроку	Дії викладача	Дії тих, хто навчається
1	Організація початку заняття	Встановлення тиші. Вітання здобувачів освіти. Відмітка присутніх.	Відповідають на вітання викладача. Готують усе необхідне приладдя для роботи. Розміщуються за комп'ютерами.
2	Повідомлення теми, цілі та мотивація	Оголошує тему уроку: «Введення у Django» та його мету: «Навчитися налаштовувати Django-проєкт, розуміти структуру файлів та запускати перший веб-додаток».	Слухають, записують тему заняття в зошити. Осмислюють значення теми для майбутньої професійної діяльності.
3	Аналіз сформованості та актуалізація знань	Методом усного опитування проводить контроль базових знань: 1. Що таке Python? 2. Для чого використовують фреймворки? 3. Що ви знаєте про Django?	Відповідають на запитання викладача, пригадуючи вивчений матеріал. Слухають коментарі викладача.
4	Формулювання та пояснення задач і завдань	Викладач оголошує задачі заняття: 1. Встановити Django. 2. Створити новий проєкт. 3. Ознайомитися зі структурою файлів Django. 4. Запустити тестовий сервер.	Усвідомлюють зміст завдань заняття. Слухають пояснення викладача та записують основні моменти в зошити.
5	Формування орієнтовної основи діяльності	Демонструє на екрані покроковий процес: 1. Встановлення Django через pip. 2. Створення проєкту командою django-admin startproject. 3. Огляд основних файлів: settings.py, urls.py, manage.py. 4. Запуск тестового сервера командою python manage.py runserver.	Слухають і спостерігають за демонстрацією. Записують основні команди та пояснення у зошити.
6	Виконавчі дії	Студенти виконують завдання	Викладач спостерігає

№ п/п	Структурні елементи уроку	Дії викладача	Дії тих, хто навчається
		на своїх комп'ютерах: встановлюють Django, створюють проєкт, досліджують структуру файлів, запускають сервер і перевіряють роботу веб-додатка.	за виконанням завдань, відповідає на запитання, допомагає вирішувати проблеми.
7	Аналіз виконаної роботи	Проводить обговорення: 1. Які труднощі виникли під час виконання завдань? 2. Що вдалося зробити найкраще? 3. Яка структура файлів у Django-проєкті?	Студенти діляться своїм досвідом виконання завдань, обговорюють результати з викладачем та одногрупниками.
8	Оцінка результатів роботи та домашнє завдання	Викладач оцінює виконані завдання, коментує типові помилки. Задає домашнє завдання: 1. Ознайомитися з документацією Django. 2. Створити додатковий проєкт із базовою структурою.	Студенти записують домашнє завдання, задають уточнюючі запитання.

5.11. Розробка одного з видів засобів навчання.

Для факультативного заняття на тему «Створення веб-додатків на Python з використанням бази даних» було розроблено презентацію, яка слугує засобом навчання та підтримкою викладача. У теоретичному блоці пояснюється основи роботи з Django, структура фреймворку, взаємодія з базою даних SQLite.

Практичний блок презентує покрокову інструкцію створення веб-додатку: налаштування середовища, проєктування бази даних, розробку моделей, шаблонів і форм. Для перевірки знань пропонуються теоретичні запитання та практичні завдання, які студенти мають виконати під час заняття.

Презентація завершується підсумками, які узагальнюють ключові аспекти роботи, а також завданнями для самостійного опрацювання. Її дизайн мінімалістичний, із використанням чітких шрифтів та прикладів коду, що допомагає візуалізувати складні концепції й зробити матеріал доступнішим.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі було розглянуто важливість професійної підготовки фахівців з комп'ютерних технологій для розробки веб-додатків на Python із використанням бази даних SQLite. Зроблено акцент на тому, що ці технології мають широке застосування в різних сферах, зокрема в автоматизації бізнес-процесів, розробці інформаційних систем, створенні мобільних додатків та веб-сервісів.

Технології Python та SQLite були детально проаналізовані з точки зору їх можливостей та інтеграції для розробки веб-додатків. Зокрема, було висвітлено роль SQLite як легкої вбудованої бази даних, яка ефективно використовує ресурси та підходить для невеликих і середніх проєктів. Окрему увагу приділено процесу інтеграції Python і SQLite в веб-розробці, а також вибору програмного забезпечення для back-end і front-end частини додатків.

Практичний приклад створення веб-додатку на Python з використанням SQLite наочно продемонстрував архітектуру веб-додатку, а також процес розробки back-end частини та інтерфейсу користувача. Розробка інструкцій для адміністраторів і кінцевих користувачів допомогла забезпечити зручність використання додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Software Foundation. *The Python Language Reference Manual*. Онлайн ресурс
2. Holovaty A., Kaplan-Moss J. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2020.
3. Owens M. *The Definitive Guide to SQLite*. Apress, 2019.
4. Krug K. *Using SQLite*. O'Reilly Media, 2020.
5. Flanagan D. *JavaScript: The Definitive Guide*. O'Reilly Media, 2020.
6. Duckett J. *HTML and CSS: Design and Build Websites*. Wiley, 2014.
7. Pollock D. *Beginning Python: Using Python 2.6 and Python 3.1*. Wrox, 2021.
8. Matthes E. *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press, 2019.
9. В. В. Черемних, О. А. Скворцова. *Програмування на Python. Практичний курс*. Видавництво Національного технічного університету України, 2021.
10. Методичні матеріали кафедри комп'ютерних наук та інформаційних технологій (локальні ресурси навчального закладу).
11. Real Python — інтерактивні уроки Python та Django.
12. Django Documentation — офіційна документація Django.
13. Бушуев С. Д., Горбачук О. В. *Використання Python у створенні веб-додатків: досвід і перспективи*. *Сучасні інформаційні технології та інноваційні методи в освіті*, 2022.
14. Smith J., Anderson P. *Modern Practices in Python Web Development*. *Journal of Software Engineering and Applications*, 2023.
15. Sweigart A. *Automate the Boring Stuff with Python*. No Starch Press, 2020.
16. Ziadé T. *Expert Python Programming*. Packt Publishing, 2019.